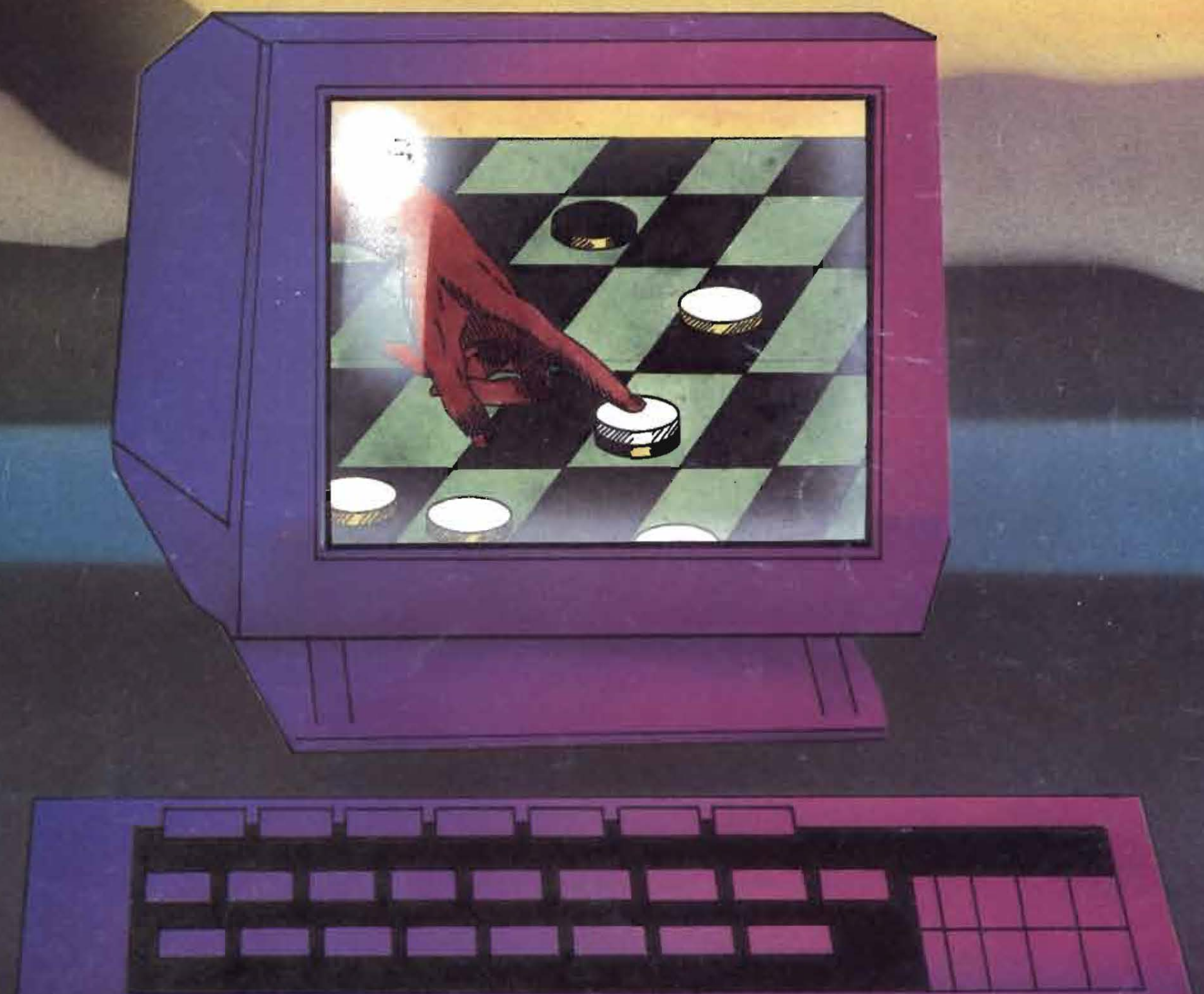
LA PRIMA RIVISTA EUROPEA DI SOFTWARE PER PERSONAL COMPUTER

ISSN 0392-8896

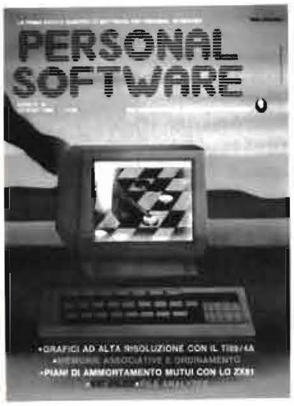
PERSONAL SOFTWARE

ANNO 3 N. 14
GENNAIO 1984 L. 3.500

UNA PUBBLICAZIONE DEL GRUPPO EDITORIALE JACKSON



- GRAFICI AD ALTA RISOLUZIONE CON IL TI99/4A
- MEMORIE ASSOCIATIVE E ORDINAMENTO
- PIANI DI AMMORTAMENTO MUTUI CON LO ZX81
- VIC PILOT • FILE ANALYZER



In copertina: raffigurazione di una partita a Dama con il computer.

N. 14
GENNAIO 1984

ARTICOLI

GUIDA

Pag.	Articolo	Guida
8	**VIC PILOT** di *Flavio Stella*	VIC 20
14	**PROBABILITA' E FREQUENZA RELATIVA** di *Riccardo Mazzurco* e *Manlio Flora*	Apple
18	**PROGRAMMER'S TOOL KIT PER COMMODORE 64 2°** di *Alessandro Guida*	C 64
27	**FILE ANALYZER PER APPLE** di *Sergio Orlando*	Apple
37	**SEMPLIFICAZIONE DI FRAZIONI PER TI 58-59** di *Francesco Carbone*	TI 58-59
41	**MEMORIE ASSOCIATIVE E ORDINAMENTO** di *Maurizio Coccorese*	generico
47	**GRAFICI AD ALTA RISOLUZIONE CON IL TI/994A** di *Sergio Borsani*	TI99/4A
53	**OTHELLO PER ZX SPECTRUM** di *Stefano Cerutti*	ZX Spectrum
56	**DAMA CINESE E MOTOCROSS** di *Claudio Driussi*	ZX81
59	**PIANI DI AMMORTAMENTO MUTUI CON LO ZX81** di *Angelo Motta*	ZX81
63	**UN ESEMPIO DI ANALISI LESSICOGRAFICA** di *Francesco Sardo*	ZX Spectrum
65	**ALCUNI TRUCCHI DELLO SPECTRUM** di *Tullio Policastro*	ZX Spectrum

RUBRICHE

Pag.	Rubrica	Guida
5	**EDITORIALE** di *Riccardo Paolillo*	
7	**POSTA**	
	I SEGRETI DEI PERSONAL:	
68	**DISABILITAZIONE LIST: LA NUOVA FUNZIONE DI APPEND** di *Italo Albanese*	VIC 20 C 64
69	**COME REALIZZARE LA FUNZIONE APPEND** di *Lucio Iacono*	VIC 20 C 64
70	**COME PROTEGGERE I VOSTRI PROGRAMMI** di *Marcello Spero*	ZX Spectrum
72	**TRE NUOVE ISTRUZIONI** di *Mauro Lenzi*	Sharp
73	**IL MOVIMENTO DEL TI BASIC** di *Sergio Borsani*	TI99/4A
77	**DEBUG**	PET-CBM
79	**PICCOLI ANNUNCI**	

L'ora di informatica

di Riccardo Paolillo

Dall'inizio di quest'anno scolastico, si sono moltiplicati i casi di introduzione di corsi di informatica nei programmi di studio in molte scuole.

Probabilmente non poteva essere altrimenti visto lo straordinario interesse con il quale vengono seguiti i veloci sviluppi di questa scienza da un numero sempre maggiore di persone.

Occorre osservare che già negli anni scorsi c'erano stati alcuni esperimenti di questo tipo in varie parti d'Italia; si trattava però, in genere, di casi isolati voluti da singoli insegnanti e presidi che con tanta passione e spesso pochi mezzi, cercavano di supplire con il loro sforzo alla mancanza di corsi istituzionalizzati.

Quest'anno, invece, sebbene manchi sempre un piano nazionale ministeriale (che sarebbe comunque, in questo momento, di problematica realizzazione), assistiamo alla nascita di corsi più organizzati, che coinvolgono organicamente varie scuole.

A Milano buona parte dei licei prevede corsi di 120 ore che coprono tematiche di base, affiancati a conferenze e dibattiti.

Ma l'informatica non entra solo nelle superiori: già il prossimo anno verrà sperimentata a Torino nelle scuole medie e addirittura esiste il progetto di portare il calcolatore nelle elementari.

Questi sono solo alcuni esempi di ciò che sta succedendo nel mondo scolastico italiano.

D'altra parte, il processo di meccanizzazione di certe attività non intellettuali, procede sicuro ed inesorabile e tra pochissimi anni doversi servire di un terminale per compiere determinate operazioni bancarie o per ricercare un libro nella biblioteca comunale, sarà assolutamente necessario per tutti noi.

Occorre quindi smitizzare il calcolatore, soprattutto per quanto riguarda le applicazioni personali, fornendo a tutti la preparazione di base che ne consenta un proficuo impiego.

L'obiettivo è quello di far sì che l'informatica diventi una materia di studio fissa per tutte le scuole dell'obbligo e superiori.

Naturalmente non sarà facile realizzarlo: esistono attualmente dei problemi reali, come i costi delle attrezzature necessarie e la reperibilità di insegnanti qualificati, che rendono di difficile attuazione questo progetto. Per ora è importante incentivare le attività attualmente in fase di sperimentazione, e cercare di promuoverne altre che spesso, grazie al patrocinio di aziende private, possono vedere la luce con stanziamenti pubblici modesti o addirittura inesistenti.

Ma soprattutto occorre evitare a tutti i costi che dopo un periodo di euforia iniziale caratterizzato da investimenti anche cospicui, tutto torni come prima e al danno economico si aggiunga la beffa di aver perso un'altra grande occasione per portare i programmi della scuola al passo con i tempi. ■



POSTA

Routine di caricamento per lo ZX Spectrum

Sono un assiduo lettore e utente dello ZX Spectrum della Sinclair. Desidererei avere delle soddisfacenti informazioni sull'uso del linguaggio macchina (Assembler) ed in particolare vorrei:

a) alcune routine per caricare il linguaggio macchina nella memoria del computer;
b) informazioni sull'input e l'output dei dati che verranno elaborati dal programma caricato;
c) avere, se possibile, alcuni programmi (in linguaggio macchina) dimostrativi.

Coretti Antonio
Matera

Esistono vari metodi per caricare routine in linguaggio macchina e alcuni sono stati illustrati su numeri precedenti della rivista.

Nel numero 8-9, nell'ambito della rubrica I segreti dei personal, il nostro collaboratore Marcello Spero ha presentato alcune routine scritte in linguaggio macchina e un piccolo programma BASIC per caricarle in memoria.

Il programma esegue ciclicamente prima la lettura mediante una istruzione READ dei byte da caricare (che ovviamente devono essere presenti in opportune istruzioni DATA) e quindi li memorizza tramite una POKE.

Un altro metodo, molto usato per introdurre brevi routine utilizzate da programmi BASIC, consiste nell'inserire all'inizio del programma una istruzione REM apparentemente senza senso, ma contenente in realtà i byte della routine linguaggio macchina da caricare. In questo modo all'inizio della zona di memoria in cui è memorizzato il programma BASIC viene automaticamente a trovarsi la nostra routine.

Questo metodo presenta il vantaggio di non richiedere il caricamento esplicito in memoria (dato che avviene in modo automatico con il LOAD del programma BASIC) e di non richiedere la protezione del programma da possibili sovrascritture (infatti in esecuzione le istruzioni REM vengono ignorate).

Anche di questa tecnica abbiamo pubblicato alcuni esempi di utilizzo: le segnalo, ad esempio, l'articolo "Rotazione bidirezionale su video dello ZX81" pubblicato nel n. 10 - 11.

Per quanto riguarda l'input e output dei dati, non dipende dalla tecnica di caricamento utilizzata, ma da come effettivamente opera la routine inserita. Tenga comunque presente che le operazioni di input e output, che normalmente sono piuttosto laboriose da scrivere in programmi BASIC, diventano ovviamente molto più complicate nei programmi in linguaggio macchina.

C 64 e musica

Ho acquistato recentemente un Commodore C 64, attratto dalle sue interessanti caratteristiche. In effetti è un ottimo calcolatore, ma ritengo che il manuale italiano di utilizzo in dotazione sia poco esauriente su alcuni importanti argomenti.

In particolare, il capitolo che tratta della possibilità di ottenere suoni è molto corto e sicuramente poco chiaro per i principianti. Io ho provato a scrivere qualche programma, ma i primi risultati non sono stati eccezionali. Avete in previsione qualche articolo che possa aiutarmi?

Marco Zetti
Milano

Il manuale in dotazione al C 64, in effetti, non rende certo giustizia alle sue notevoli possibilità. Stranamente vengono minimizzate o addirittura taciute alcune delle caratteristiche più interessanti e innovative.

Da qualche tempo è comunque disponibile presso i rivenditori autorizzati la "Guida al C 64" in versione italiana, ottimo e corposo testo veramente esauriente. Speriamo, per il futuro, che venga consegnato gratuitamente agli acquirenti del C 64, in quanto costituisce un utile se non indispensabile mezzo per sfruttare al meglio il calcolatore.

Per quanto ci riguarda, stiamo preparando una serie di articoli su applicazioni musicali con C 64, che pubblicheremo molto presto.

Le ricordo che nel n. 40 di **Bit** *sono stati pubblicati due articoli di Mirko Gremes su questo argomento.*

Inverse video Spectrum

Vi scrivo per porvi alcune domande riguardo lo Spectrum della Sinclair.

In particolare vorrei conoscere lo scopo dei comandi "True video" e "Inverse video" in quanto essi non vengono trattati dal manuale.

Greggio Daniele
Novara

L'effetto del comando Inverse video, che va comunque inserito in una striga da stampare, è quello di invertire la matrice dei punti di tutti i caratteri che seguono.

Il ripristino si ottiene ovviamente utilizzando il comando True video. Il comando Inverse può anche essere inserito nelle linee di programma per ottenere la stampa in campo inverso del listato.

PERSONAL SOFTWARE

VIC Pilot

Anche sul VIC 20 questo linguaggio che permette la programmazione della grafica ad alta risoluzione in modo semplice e lineare

di *Flavio Stella*

I linguaggi Pilot e Logo stanno ottenendo un successo considerevole per le loro possibilità logiche e grafiche tra gli appassionati ed anche nel campo professionale dell'educazione.

Questi due linguaggi sono incentrati sulla gestione grafica di una immaginaria "tartaruga" che muovendosi lascia una traccia delle sue evoluzioni sullo schermo formando così il disegno in alta risoluzione che costituisce l'oggetto principale del programma.

È necessaria l'espansione minima di 8 Kbyte per poter caricare in memoria l'interprete Pilot e lasciare uno spazio sufficiente ai programmi veri e propri.

La logica Turtle Graphic

Il metodo grafico, a tutti noto ed usato nell'insegnamento della matematica analitica, è basato sull'utilizzo delle coordinate cartesiane x e y che determinano univocamente la posizione di un punto nel piano, permettendo così di descrivere linee e figure come insieme dei punti aventi le coordinate soddisfacenti una espressione algebrica che viene chiamata equazione.

L'approccio Turtle Graphic è sostanzialmente più istintivo e pertanto più adatto ad un utilizzo amatoriale ed educativo. Immaginandosi di camminare al posto del simpatico animale, il programmatore deve descrivere i movimenti e le distanze a partire dalla sua posizione attuale anzichè da un generico punto detto origine.

Un quadrato sarà descritto da un cammino di n passi ed un cambiamento di direzione di 90° ripetuti quattro volte come se si stesse passeggiando sul perimetro di un'aiuola quadrata.

Con questo metodo è possibile avvicinare i bambini (anche delle prime classi elementari) all'uso del calcolatore che nel contempo fornisce loro i mezzi per l'acquisizione di concetti geometrici e topografici di notevole utilità.

L'interprete Pilot

L'interprete è quell'insieme di istruzioni che permette di trasformare un set di parole e regole sintattiche in azioni elementari gestite dal microprocessore. Le istruzioni che definiscono l'interprete devono realizzare le intenzioni del programmatore in modo corretto e rispettando le priorità assegnate.

Il programma (listato 2), scritto completamente in BASIC, permette l'utilizzo di tre gruppi di istruzioni: dirette, di input/output e logiche, grafiche.

L'interprete Pilot controlla la posizione, la direzione ed il movimento della tartaruga condizionandola alle istruzioni del programma e alle eventuali informazioni ricavate dalla tastiera, da confronti relazionali, da calcoli elementari.

La grafica è costruita su un quadro di 160x176 pixel e, con l'espansione minima richiesta di 8 Kbyte, rimangono 2 Kbyte per i programmi Pilot; il risultato che si ottiene è un disegno tracciato punto per punto, con un metodo familiare a chi ha già lavorato con il VIC in alta risoluzione, nitido e simpatico ma, purtroppo, un po' lento. È possibile modificare il programma per ottenere una maggiore velocità di esecuzione se si possiede la cartuccia Super-Expander.

L'editor Pilot

L'editor è quella serie di istruzioni che codifica e immagazzina nella memoria le righe del programma permettendo poi le modifiche e le correzioni che si rendessero necessarie. Non vi sono differenze sostanziali con l'editor del BASIC, comunque è opportuno ricordare che:

- le correzioni si possono fare in qualsiasi punto della riga e che il tasto RETURN inserisce la riga corretta in memoria al posto della versione precedente;
- la cancellazione di una riga si ottiene digitando il numero della riga ed il tasto RETURN cioè memorizzando una riga nulla;
- ogni riga di istruzioni non preceduta da un numero viene considerata un comando diretto.

Istruzioni operative

Essendo tutta la memoria interna del VIC utilizzata per creare lo schermo ad alta risoluzione l'inizio del programma BASIC deve essere spostato alla locazione 8192 *prima di caricare in memoria il programma e lanciarlo*; se questa operazione non viene eseguita, il programma si autocancellerà in parte creando in-

spiegabili vuoti e messaggi di errore.

Ogni volta quindi che questo programma viene caricato è necessario inserire nuovi valori nei puntatori di inizio BASIC per mezzo della linea seguente:

POKE44,32:POKE642,32:POKE 8192,0:NEW

Per interrompere l'esecuzione di un programma basta premere il tasto @ che restituisce lo schermo normale appena la linea in corso di esecuzione è completata. Se si tratta, però, di un processo iterattivo molto lungo il ritardo può essere rilevante; in questo caso, o quando si è creata una routine che richiama se stessa senza fine, sarà indispensabile tornare al Pilot attraverso il BASIC premendo RUN/STOP e RESTORE e poi digitando GOTO11. Così facendo il programma non verrà cancellato e sarà a disposizione per ripartire o per le modifiche necessarie.

Nell'operazione di LOAD il programma caricato in memoria dal nastro viene fuso con quello eventualmente esistente in memoria; per ovviare ai problemi che ne potrebbero derivare è opportuno ricordarsi di eseguire sempre un NEW prima di ogni LOAD. Questo inconveniente può essere trasformato in vantaggio se si vuole fondere una o più routine programmate separatamente in un unico programma; per ottenere questo risultato sarà necessario dare numerazione diversa alle righe delle singole sottoprocedure per evitare le sovrapposizioni. Per destinare i comandi Load e Save ad una unità floppy, modificare il programma come segue:

```
41 OPEN1,8,2,R$+'',S,W-
   '':PRINT
   ''SAVING''R$
```

I$	*Accumula i caratteri ricavati dalla routine in LM sino al RETURN.*
M	*N° massimo delle linee di programma (può essere modificato a piacere).*
C$(n)	*Vettore istruzioni Pilot.*
G$(n)	*Vettore istruzioni grafiche.*
B$(n)	*Vettore argomenti delle istruzioni PEN, BORDER, SCREEN.*
L	*N° di linea Pilot.*
L$(k)	*Istruzioni contenute nella riga k.*
N%(y)	*Vettore delle variabili numeriche y = [(valore ASCII della lettera)-64].*
S$(y)	*Vettore delle variabili alfanumeriche y = [(valore ASCII della lettera)-64].*

Figura 1. *Elenco delle principali variabili utilizzate nel programma.*

1- 3	*Gestisce la routine in LM (caricata nel buffer della cassetta) che legge le istruzioni in fase di editing.*
4-10	*Inizializza i vettori contenenti le parole Pilot.*
11	*Messaggio Pilot che conclude tutte le operazioni come READY.*
12	*Legge l'input, se blank ripete.*
13	*Elimina gli spazi che precedono il numero di riga o il comando.*
14-22	*Gestisce i comandi diretti.*
23-30	*Interpreta le istruzioni di programma.*
31-50	*Routine di esecuzione dei comandi diretti (vedi tabella 2).*
51-60	*Esegue il programma (RUN).*
61	*Messaggio di errore (per codici vedi tabella 1).*
62	*Routine di esecuzione delle istruzioni Pilot (vedi tabella 2).*
116-126	*Riconoscimento delle istruzioni grafiche.*
127	*Ritorna alla visualizzazione del testo per permettere la segnalazione degli errori o per fine procedura grafica.*
130-138	*Esecuzione del comando iterativo DRAW+TURN.*
139	*Routine grafiche (vedi tabella 2).*

Figura 2. *Descrizione delle principali routine del programma VIC Pilot.*

```
45 OPEN1,8,2,R$+'',S,R-
   '':PRINT
   ''LOADING''R$
```

Qualsiasi infrazione alle regole Pilot o ai limiti imposti genererà un messaggio d'errore contenente il numero di linea ed un codice (vedi tabella 1).

Infine è da tener presente che l'interprete non è indifferente agli spazi posti fra le istruzioni; gli spazi prima del numero di linea e immediatamente dopo vengono rimossi automaticamente, ma in tutti gli altri casi la presenza o l'assenza di uno spazio può generare un messaggio d'errore.

Comandi diretti

LIST xx-yy	Esegue il listing del programma tra le linee in argomento; uno od entrambi i numeri di linea possono essere omessi.
RUN	Esegue il programma Pilot in memoria.
SAVE nome	Registra il programma in memoria sulla cassetta.
LOAD nome	Carica in memoria il programma dalla cassetta.
NEW	Cancella ogni istruzione presente in memoria.
BASIC	Restituisce il controllo al BASIC.
PLIST xx-yy	Come il comando LIST ma con output su stampante.

Ognuno di questi comandi può

essere accorciato anche ad una sola lettera:
Esempio: L 10-25 lista il programma tra le linee 10 e 25.

Variabili

L'interprete riconosce come variabili alfanumeriche, quelle composte dal segno $ seguito da una lettera, come variabili numeriche intere, quelle precedute dal segno #.

Istruzioni Pilot

Le frasi Pilot, escluse le LABEL, consistono di: un nome, un condizionatore (facoltativo), il segno ":" ed un argomento (numerico, alfanumerico o variabile).
T: = TYPE Stampa quello che è contenuto nell'argomento sullo schermo, sia esso testo o variabile; se la linea termina con ";" non va a capo.
Esempio: 10 T:ANGOLO= # A stamperà ANGOLO=xx dove xx è il valore della variabile numerica intera # A.
A: = ACCEPT Riceve una informazione dall'utente per mezzo della tastiera, l'argomento può essere una variabile ma non è indispensabile, nel caso sia assente l'input viene immagazzinato in un buffer che può essere consultato con l'istruzione MATCH.
Esempio: 15 A: - 20 A:$V
M: = MATCH Verifica se una o più stringhe in argomento sono contenute in una variabile alfanumerica oppure nel buffer di ACCEPT se questa non è presente; se la verifica dà risultato positivo viene messo a 1 il flag Y (yes) altrimenti il flag N (not).
Esempio: 15 M:12,DODICI-,XII,1100 mette a 1 il flag Y se una di queste rappresentazioni di dodici è nel buffer di ACCEPT 20 M:$L,GARDA,COMO,ISEO dà Y=1 se uno dei nomi di laghi è contenuto nella variabile $L.
I: = IF L'istruzione IF non è normalmente inclusa in questo linguaggio ma è implementata in questo programma per permettere confronti tra variabili con gli operatori relazionali (<,>,=); il risultato sarà rilevabile dal valore dei flag Y e N.
Esempio:
30 I: # N=9
35 I: # N< # L
J: = JUMP
U: = USE Corrispondono a GOTO e GOSUB però il loro argomento può essere sia un numero di linea che una LABEL.
Esempio: 35 J:5
40 U:★INIZIO
E: = END Corrisponde al RETURN perchè chiude una subroutine e riprende dalla riga successiva all'istruzione USE.
C: = COMPUTE Questa istruzione calcola semplici espressioni lineari e senza parentesi fino ad un massimo di quattro operazioni, l'argomento deve essere un'equazione il cui risultato viene posto nella variabile numerica a sinistra dell'uguale.
Esempio:

15 C: # N= # G★10/# T+15
La variabile # R incontrata in un'espressione sarà considerata come valore casuale compreso tra 0 e 1.
R: = REMARK Precede un commento.
H: = HOME Cancella il testo e posiziona il cursore in alto a sinistra.
G: = GRAPHIC In argomento contiene uno dei comandi della grafica elencati nel paragrafo successivo.
END Viene posto a concludere la esecuzione di un programma Pilot e non può essere abbreviato né soggetto a condizione come le altre istruzioni.
CONDIZIONATORI Tutte le istruzioni Pilot tranne END possono essere condizionate ai due flag Y e N che sono predisposti con le istruzioni MATCH e IF.
Esempio: 10 TY:PROVA stampa PROVA se il flag Y = 1
15 JN:★INIZIO salta a ★INIZIO se il flag N = 1
LABEL Servono ad identificare linee di programma per le istruzioni Jump e USE senza ricorrere al numero di linea che può non essere ancora noto, sono costituite da un nome qualsiasi preceduto da un asterisco.
Esempio:
10★INIZIO
20 ....
30 ....
40 JY:★INIZIO
50 ....

1	*Variabile con nome non ammesso.*
2	*Label sconosciuta.*
3	*Troppe subroutine richiamate contemporaneamente (max. 8).*
4	*Istruzione E: incontrata alla fine di una subroutine non precedentemente richiamata.*
5	*Errore di sintassi.*
6	*Divisione per 0.*
7	*Variabile numerica fuori dall'intervallo ammesso (>|32767|).*
8	*Routine grafica non preceduta dall'istruzione CLEAR.*

Tabella 1. *Codificazione dei messaggi d'errore.*

32	*LIST*	62	*TYPE*	139	*CLEAR*
51	*RUN*	73	*JUMP*	142	*QUIT*
41	*SAVE*	76	*END*	145	*TURN*
45	*LOAD*	71	*USE*	147	*DRAW*
49	*NEW*	78	*MATCH*	157	*GO*
50	*BASIC*	85	*COMPUTE*	158	*PEN*
31	*PLIST*	101	*ACCEPT*	163	*SCREEN*
		106	*IF*	165	*BORDER*
		115	*HOME*		
		116	*GRAPHIC*		
			REMARK non si esegue		

Tabella 2. ***Linee del programma BASIC dove si trovano le routine che eseguono funzioni Pilot.***

VIC Pilot

Istruzioni grafiche

Tutte le istruzioni grafiche illustrate in questo paragrafo devono essere precedute da G: come risulta da tutti gli esempi forniti.

CLEAR Inizializza e cancella lo schermo ad alta risoluzione sistemando fondo bianco, bordo blu e traccia nera; la tartaruga viene sistemata al centro del quadro orientata a zero gradi, nord, questo comando *deve* essere il primo di ogni routine grafica.
Esempio: G:CLEAR oppure G:C

TURN Può essere seguito da un numero o da una variabile che rappresentano il valore in gradi che viene aggiunto alla direzione attuale, la rotazione è oraria con angoli positivi e negativi.
Esempio: G:T # A oppure G:T 90

TURNTO Simile al precedente ma la direzione è fissata all'angolo in argomento in modo assoluto cioè non considerando la posizione attuale; qualsiasi abbreviazione deve comunque contenere il suffisso TO.
Esempio: G:TURNTO 90 oppure G:TTO # A

DRAW Muove la tartaruga per la distanza fissata nell'argomento lasciando una traccia se la "penna è poggiata" (vd. PEN); se il disegno esce dal quadro visibile si genera un messaggio di avvertimento visualizzato al ritorno al "text mode".
Esempio: G:DRAW # A oppure G:D 120

GOTO Posiziona la tartaruga con coordinate cartesiane che possono oscillare nell'intervallo [$-106.65 \leqslant x \leqslant 108$ e $-87 \leqslant Y \leqslant 88$].
Esempio: G:GOTO # X, # Y oppure G:GTO − 15,35

GO Come DRAW ma non lascia la traccia, come DRAW e PEN UP.

PEN Controlla il colore della traccia lasciata dalla tartaruga sullo schermo; i colori ammessi sono BLACK, WHITE, RED, CYAN PURPLE, GREEN, BLUE, YELLOW; se PEN è seguito da ERASE il colore selezionato è quello del fondo cosicchè ripassando su un segno già presente si ottiene l'effetto di cancellarlo; PEN seguito da UP disinserisce la traccia nelle istruzioni DRAW che seguono, con PENDOWN si riabilita la scrittura; *nessuno dei suffissi di PEN può essere abbreviato.*
Esempio: G:P GREEN

SCREEN Cambia il colore dello schermo (sfondo) senza cancellare i disegni già realizzati; è seguito da uno dei colori visti per PEN.
Esempio: G:S RED

BORDER Come SCREEN ma opera sul bordo.

QUIT Conclude la routine grafica e restituisce lo schermo normale; per permettere l'osservazione dei disegni creati per un tempo a piacere questa istruzione è inibita con una GET che richiama se stessa sino alla pressione del tasto "Q"; QUIT deve essere l'ultima istruzione di ogni procedura grafica altrimenti, per interrompere il programma, si è costretti ad abortire la procedura premendo il tasto @.

DRAW+TURN È spesso necessario ricorrere ad una serie di DRAW e TURN consecutivi, soprattutto disegnando poligoni regolari, l'interprete Pilot permette di semplificare e abbreviare il programma con la forma: G:xx(DRAW yy;TURN zz); xx deve essere un numero intero mentre yy e zz possono essere numeri o variabili a seconda delle esigenze.
Esempio: G:4(D # A;T 90) —— quadrato di lato # A ———
G:3(D # A;T 60) ——triangolo equilatero.

Conclusioni

Un programma dimostrativo scritto in Pilot è riportato nel listato 1.

L'obiettivo del programma VIC Pilot è quello di permettere l'uso della Turtle Graphic sul VIC 20 con solo 8 Kbyte di memoria addizionale. Sacrificando parte della memoria disponibile per i programmi o disponendo di maggiore capacità, sarebbe auspicabile un ampliamento del set di istruzioni soprattutto per rendere possibile il comando del disegno in modo interattivo con la fusione di immagini grafiche e testo; sarebbe anche utile poter utilizzare il suono per rallegrare l'attesa. Il programma si presta, in ogni caso, ad essere arricchito e nel contempo a ricompensare qualsiasi approfondimento con una migliore comprensione dei meccanismi apparentemente "automatici" che stanno dietro lo svolgimento di tutti i nostri programmi. Per facilitare il compito a chi volesse arricchire l'interprete Pilot, la tabella 2 fornisce le linee del programma BASIC che codificano le varie istruzioni Pilot e la figura 1 e 2 contengono rispettivamente la lista delle principali variabili e l'elenco delle principali routine. ■

Listato 1. *Questo programma Pilot è stato scritto per dimostrare praticamente cosa è possibile realizzare e nel contempo provare l'efficienza del vostro nuovo linguaggio VIC Pilot.*

```
1 G:C
3 G:GTO -15,30
5 C:#A=1
6 C:#B=16
8 G:S WHITE
10 G:B RED
12 U:*COLOR
14 G:TTO #B
20 U:*STAR
22 G:T 90
24 G:P UP
26 G:D 25
28 G:P DOWN
30 C:#A=#A+1
32 C:#B=#B+72
34 I:#B>360
36 JN:12
38 J:80
40 *STAR
42 G:5(D 20;T 144)
44 E:
50 *COLOR
51 I:#A=1
52 JY:61
53 I:#A=2
```

VIC Pilot

Seguito listato 1.

```
54 JY:63
55 I:#A=3
56 JY:65
57 I:#A=4
58 JY:67
59 I:#A=5
60 JY:69
61 G:P BLUE
62 J:70
63 G:P CYAN
64 J:70
65 G:P RED
66 J:70
67 G:P GREEN
68 J:70
69 G:P PURPLE
70 E:
80 G:TTO 18
81 G:P BLACK
82 G:GTO -50,-70
83 G:D 40
84 G:T 60
85 G:20(D 2;T 12)
86 G:GTO -25,-50
88 G:TTO 198
90 G:D 3
91 G:GO 3
92 G:D 9
93 G:18(D 1;T -11)
94 G:GTO -5,-40
96 G:TTO 198
97 G:D 26
98 G:18(D 1;T -11)
100 G:P GREEN
101 G:GTO 13,-57
102 G:TTO -90
106 G:11(D 2;T -15)
107 G:T -20
108 G:10(D 2;T -15)
110 G:P BLACK
111 G:GTO 34,-40
112 G:TTO 198
114 G:1(D 6;T -108)
116 G:2(D 5;T 180)
117 G:TTO 198
118 G:D 20
120 G:18(D 1;T -11)
121 G:P BLUE
122 G:TTO 0
123 G:GTO -75,-77
124 G:1(D 140;T 90)
126 G:1(D 140;T 90)
128 G:D 140
130 G:QUIT
131 END
```

Listato 2. *Programma interprete Pilot.*

```
 0 GOTO4
 1 I$=""
 2 SYS820:IFPEEK(0)=13THENRETURN
 3 I$=I$+CHR$(PEEK(0)):GOTO2
 4 POKE36866,150:POKE36869,240:POKE648,
30
 5 FORJ=217TO228:POKEJ,158:NEXT:FORJ=22
9TO250:POKEJ,159:NEXT
 6 CLR:M=200:DIMS%(9),N%(26),S$(26),L$(
M),C$(17),G$(7),B$(10)
 7 PRINT"■ **** PILOT V2.1 ****":FORX=
820TO825:READZ:POKEX,Z:NEXT:FORX=0TO17
 8 READC$(X):NEXT:FORX=0TO7:READG$(X):N
EXT:FORX=0TO10:READB$(X):NEXT:DATA32,20
7,255,133
 9 DATA0,96,LIST,RUN,SAVE,LOAD,NEW,BASI
C,PLIST,T,J,E,U,M,C,A,I,H,R,G,CLEAR,QUI
T,TURN
 10 DATADRAW,GO,PEN,SCREEN,BORDER,BLACK
,WHITE,RED,CYAN,PURPLE,GREEN,BLUE,YELLO
W,ERASE,UP
 11 PRINT"■PILOT.":DATADOWN
 12 GOSUB1:PRINT:IFASC(I$)=32ANDLEN(I$)
=1THEN12
 13 IFLEFT$(I$,1)=" "THENI$=MID$(I$,2):
GOTO13
 14 L=VAL(I$):IFL<>0THEN23
 15 L=1:H=M:R$="":FORX=1TOLEN(I$):IFMID
$(I$,X,1)<>" "THENNEXT:GOTO21
 16 R$=MID$(I$,X+1):I$=LEFT$(I$,X-1)
 17 L=VAL(R$):H=L:FORX=1TOLEN(R$):IFMID
$(R$,X,1)<>"-"THENNEXT:GOTO19
 18 L=VAL(LEFT$(R$,X-1)):H=VAL(MID$(R$,
X+1))
 19 IFL=0THENL=1
 20 IFH=0THENH=M
 21 FORX=0TO6:IFI$<>LEFT$(C$(X),LEN(I$)
)THENNEXT:PRINT"UNKNOWN COMMAND.":GOTO1
1
 22 ONX+1GOTO32,51,41,45,49,50,31
 23 IFL>MTHENPRINT"LINE NUMBER OUT OF
  RANGE.":GOTO11
 24 X=LEN(STR$(L)):X$=MID$(I$,X):IFX$="
"THENL$(L)="":GOTO12
 25 IFLEFT$(X$,1)=" "THENX$=MID$(X$,2):
GOTO25
 26 X=3:IFMID$(X$,2,1)<>":"THENX=4:IFMI
D$(X$,3,1)<>":"THENL$(L)=X$:GOTO12
 27 FORZ=7TO17:IFLEFT$(X$,1)<>C$(Z)THEN
NEXT:PRINT"ILLEGAL COMMAND.":GOTO11
 28 IFMID$(X$,2,1)="Y"THENZ=Z+13
 29 IFMID$(X$,2,1)="N"THENZ=Z+26
 30 L$(L)=CHR$(Z-6)+MID$(X$,X):GOTO12
 31 OPEN1,4:GOTO33
 32 OPEN1,3
 33 FORX=LTOH:IFL$(X)=""THEN39
 34 X$=":":Z=ASC(L$(X)):IFZ>40THENX$=LE
FT$(L$(X),1):GOTO38
 35 IFZ>26THENZ=Z-26:X$="N"+X$
 36 IFZ>13THENZ=Z-13:X$="Y"+X$
 37 X$=C$(Z+6)+X$
 38 PRINT#1,X;X$;MID$(L$(X),2)
 39 GETX$:IFX$<>""THENCLOSE1:GOTO11
 40 NEXT:CLOSE1:GOTO11
 41 OPEN1,1,1,R$:PRINT"SAVING "R$
 42 FORX=1TOM:IFL$(X)=""THEN44
 43 PRINT#1,X;CHR$(13)CHR$(34)L$(X)CHR$
(34)CHR$(13);
 44 NEXTX:CLOSE1:GOTO11
 45 OPEN1,1,0,R$:PRINT"LOADING "R$
 46 INPUT#1,X:IFSTTHEN48
 47 INPUT#1,L$(X):IFST=0THEN46
 48 CLOSE1:GOTO11
 49 GOTO6
 50 PRINT"■EXITING TO BASIC....":END
 51 L=0:FORX=1TO26:N%(X)=0:S$(X)="":NEX
T:P=0:F%=0
 52 L=L+1:IFL=>MORL$(L)="END"THEN11
 53 GETX$:IFX$="@"ANDCO%=0THEN11
 54 IFX$="@"ANDCO%=1THENGOSUB127:GOTO11
 55 IFL$(L)=""THEN52
 56 X=ASC(L$(L)):IFX>40THEN52
 57 IFX>26THENX=X-26:IFF%=1THEN52
 58 IFX>13THENX=X-13:IFF%=0THEN52
 59 C$=MID$(L$(L),2)
 60 ONXGOTO62,73,76,71,78,85,101,106,11
5,52,116
 61 PRINT"ERROR #"E"IN LINE"L:GOTO11
 62 Z=0:IFRIGHT$(C$,1)=";"THENZ=1:C$=LE
FT$(C$,LEN(C$)-1)
 63 FORX=1TOLEN(C$):X$=MID$(C$,X,1):IFX
$="#"THEN67
 64 IFX$="$"THEN68
 65 PRINTX$;:NEXT:IFZ=0THENPRINT
 66 GOTO52
 67 GOSUB69:X$=STR$(N%(Y)):GOTO65
```

```
 68 GOSUB69:X$=S$(Y):GOTO65
 69 X=X+1:Y=ASC(MID$(C$,X,1))-64:IFY<1O
RY>26THENE=1:GOTO61
 70 RETURN
 71 IFP>8THENE=3:GOTO61
 72 P=P+1:S%(P)=L
 73 IFVAL(C$)<>0THENL=VAL(C$)-1:GOTO52
 74 FORX=1TOM:IFC$<>L$(X)THENNEXT:E=2:G
OTO61
 75 L=X:GOTO52
 76 IFP=0THENE=4:GOTO61
 77 L=S%(P):P=P-1:GOTO52
 78 X=1:C$=C$+",":X$=AC$:IFLEFT$(C$,1)=
"$"THENGOSUB83
 79 FORZ=XTOLEN(C$):IFMID$(C$,Z,1)<>","
THENNEXT
 80 Z$=MID$(C$,X,Z-X):FORY=1TOLEN(X$):I
FMID$(X$,Y,LEN(Z$))=Z$THENF%=1:GOTO52
 81 NEXT:IFZ<LEN(C$)THENX=Z+1:GOTO79
 82 F%=0:GOTO52
 83 Y=ASC(MID$(C$,2))-64:IFY<1ORY>26THE
NE=1:GOTO61
 84 X$=S$(Y):X=4:RETURN
 85 A=3:Z=0:X$="":IFLEFT$(C$,1)<>"#"ORM
ID$(C$,3,1)<>"="THENE=5:GOTO61
 86 Y=1:X$=MID$(C$,A,1):A=A+1:IFMID$(C$
,A,1)="-"THENA=A+1:Y=-1
 87 IFMID$(C$,A,1)<>"#"THENY=Y*VAL(MID$
(C$,A)):A=A+LEN(STR$(Y))-1:GOTO91
 88 X=ASC(MID$(C$,A+1))-64:IFX<1ORX>26T
HENE=1:GOTO61
 89 IFX=18THENY=Y*RND(1):GOTO91
 90 Y=Y*N%(X):A=A+2
 91 IFX$="="THENZ=Y
 92 IFX$="-"THENZ=Z-Y
 93 IFX$="+"THENZ=Z+Y
 94 IFX$="/"ANDY=0THENE=6:GOTO61
 95 IFX$="*"THENZ=Z*Y
 96 IFX$="/"THENZ=Z/Y
 97 IFA<=LEN(C$)THEN86
 98 X=ASC(MID$(C$,2))-64:IFX<1ORX>26THE
NE=1:GOTO61
 99 IFZ>32767ORZ<-32767THENE=7:GOTO61
 100 N%(X)=Z:GOTO52
 101 IFC$=""THENGOSUB1:AC$=I$:PRINT:GOT
O52
 102 X=ASC(MID$(C$,2))-64:IFX<1ORX>26TH
ENE=1:GOTO61
 103 GOSUB1:Z=VAL(I$):PRINT:IFLEFT$(C$,
1)="#"THENN%(X)=Z
 104 IFLEFT$(C$,1)="$"THENS$(X)=I$
 105 GOTO52
 106 IFLEFT$(C$,1)<>"#"THENE=5:GOTO61
 107 X=ASC(MID$(C$,2))-64:IFX<1ORX>26TH
ENE=1:GOTO61
 108 A=N%(X):X$=MID$(C$,3,1):IFMID$(C$,
4,1)<>"#"THENX=VAL(MID$(C$,4)):GOTO111
 109 X=ASC(MID$(C$,5))-64:IFX<1ORX>26TH
ENE=1:GOTO61
 110 X=N%(X)
 111 F%=0:IFX$="<"ANDA<XTHENF%=1
 112 IFX$=">"ANDA>XTHENF%=1
 113 IFX$="="ANDA=XTHENF%=1
 114 GOTO52
 115 PRINT"□";:GOTO52
 116 Y=0:FORZ=1TOLEN(C$):IFMID$(C$,Z,1)
<>" "THENNEXT:GOTO121
 117 R$=MID$(C$,Z+1):C$=LEFT$(C$,Z-1):I
FRIGHT$(C$,2)="TO"THENY=1:C$=LEFT$(C$,L
EN(C$)-2)
 118 FORZ=1TOLEN(R$):X$=MID$(R$,Z,1):IF
X$<>","ANDX$<>";"THENNEXT:GOTO121
 119 IFX$=";"THEN130
 120 X$=LEFT$(R$,Z-1):R$=MID$(R$,Z+1)
 121 FORZ=0TO7:IFC$<>LEFT$(G$(Z),LEN(C$
))THENNEXT:GOTO126
 122 IFCO%=0ANDZ<>0THENE=8:GOTO61
 123 IFZ=2ANDY=1THEN167
 124 IFZ=4ANDY=1THEN168
 125 ONZ+1GOTO139,142,145,147,157,158,1
63,165
 126 GOSUB127:PRINT"UNKNOWN GRAPHICS":P
RINT"COMMAND IN LINE ";L:GOTO11
 127 CO%=0:POKE36864,10:POKE36866,150:P
OKE36867,46:POKE36869,240:POKE36879,27
 128 IFOS%=1THENPRINT"*PLOT WENT OFF SC
REEN"
 129 PRINT"□■";:RETURN
 130 D=VAL(C$):Y=LEN(STR$(D)):C$=MID$(C
$,Y+1):IFD<=0THEN126
 131 IFC$<>LEFT$(G$(3),LEN(C$))THEN126
 132 FORZ=1TOLEN(R$):IFMID$(R$,Z,1)<>";
"THENNEXT:GOTO126
 133 C$=LEFT$(R$,Z-1):X$=MID$(R$,Z+1)
 134 FORZ=1TOLEN(X$):IFMID$(X$,Z,1)<>"
"THENNEXT:GOTO126
 135 R$=MID$(X$,Z+1):X$=LEFT$(X$,Z-1):I
FX$<>LEFT$(G$(2),LEN(X$))THEN126
 136 X$=R$
 137 R$=C$:GOTO147
 138 R$=X$:GOTO145
 139 CO%=1:UD%=0:OS%=0:POKE36864,12:POK
E36866,148:POKE36867,23
 140 POKE36869,252:POKE36879,30:CO=0:SC
=2:BC=6:AN=0:X0=0:Y0=0
 141 FORI=0TO219:POKE7680+I,I:NEXT:FORI
=4096TO7615:POKEI,0:NEXT:GOTO52
 142 GETX$:IFX$<>"Q"THEN142
 143 REM
 144 GOSUB127:GOTO52
 145 GOSUB169:AN=AN+Z:D=D-1:IFD>0THEN13
7
 146 D=0:GOTO52
 147 GOSUB169:IFZ<0THEN126
 148 TH=(90-AN)*3.1415926/180
 149 FORY=0TOZ:XG=X0+Y*COS(TH):YG=Y0+Y*
SIN(TH):IFUD%=0THENGOSUB152
 150 NEXT:X0=XG:Y0=YG:IFD>0THEN138
 151 GOTO52
 152 U=INT((XG+106.65)/1.35+.5):V=88-IN
T(YG+.5)
 153 CH=INT(V/16)*20+INT(U/8):RO=(V/16-
INT(V/16))*16
 154 IFCH<0ORCH>220ORXG<-106.65ORXG>108
THENOS%=1:RETURN
 155 BY=4096+16*CH+RO:BI=7-(U-INT(U/8)*
8)
 156 POKE38400+CH,CO:POKEBY,PEEK(BY)OR(
2↑BI):RETURN
 157 GOSUB169:TH=(90-AN)*3.14159265/180
:X0=X0+Z*COS(TH):Y0=Y0+Z*SIN(TH):GOTO52
 158 FORZ=0TO10:IFR$<>B$(Z)THENNEXT:GOT
O126
 159 IFZ<8THENCO=Z:GOTO52
 160 IFZ=8THENCO=SC-1:GOTO52
 161 IFZ=9THENUD%=1:GOTO52
 162 IFZ=10THENUD%=0:GOTO52
 163 FORZ=0TO7:IFR$<>B$(Z)THENNEXT:GOTO
126
 164 SC=Z+1:POKE36879,SC*16+BC-8:GOTO52
 165 FORZ=0TO7:IFR$<>B$(Z)THENNEXT:GOTO
126
 166 BC=Z:POKE36879,SC*16+BC-8:GOTO52
 167 GOSUB169:AN=Z:GOTO52
 168 GOSUB169:Y0=Z:R$=X$:GOSUB169:X0=Z:
GOTO52
 169 Z=VAL(R$):IFZ<>0ORR$="0"THEN173
 170 IFLEN(R$)<>2ORLEFT$(R$,1)<>"#"THEN
E=1:GOSUB127:GOTO61
 171 Y=ASC(RIGHT$(R$,1))-64:IFY<0ORY>26
THENE=1:GOSUB127:GOTO61
 172 Z=N%(Y)
 173 RETURN
```

Lista simboli grafici

```
     7 : 1    □ = SHIFT HOME   [CHR$(147)]
         1    ■ = CTRL 1   [CHR$(144)]

    11 : 1    ▣ = CRSR↑   [CHR$(17)]

    50 : 1    ▣ = CRSR↑   [CHR$(17)]

   115 : 1    □ = SHIFT HOME   [CHR$(147)]

   129 : 1    □ = SHIFT HOME   [CHR$(147)]
         1    ■ = CTRL 1   [CHR$(144)]
```

PERSONAL SOFTWARE

Probabilità e frequenza relativa

Un programma per Apple II che esemplifica i concetti esposti

di *Riccardo Mazzurco* e *Manlio Flora*

Spesso nello studio di alcune materie scientifiche viene privilegiato l'aspetto teorico di alcuni argomenti rispetto alle esemplificazioni pratiche.

Ciò impedisce che venga compreso a pieno il legame che esiste tra i fenomeni reali ed i modelli matematici che li rappresentano.

Nel nostro caso ci siamo occupati di alcuni aspetti fondamentali della teoria della probabilità.

Tale teoria è nata dalla necessità di valutare qualitativamente gli eventi casuali e pronosticarne l'andamento; le nozioni basilari di cui ci occuperemo sono quelle di frequenza relativa e di probabilità.

La frequenza relativa F(E) di un evento casuale E è data dal rapporto fra il numero N di volte in cui tale evento si è verificato, in un insieme di M prove in ciascuna delle quali avrebbe potuto verificarsi, ed il numero M stesso, cioè:

$$F(E) = N/M$$

Come criterio di valutazione di probabilità, visto che ci troviamo in presenza di un evento ripetibile, possiamo dare il seguente:
assumiamo come probabilità il limite cui tende la frequenza relativa di successo al divergere del numero di prove (supposto che tale limite esista).

Diamo un semplice esempio di tutto ciò:
come è intuitivo lanciando una monetina, non abbiamo alcun motivo per dire a priori che il risultato sarà più facilmente testa o croce; in termini probabilistici, ciò significa che i due eventi, testa e croce, sono equiprobabili, cioè:

$$P(testa) = P\ (croce) = 1/2$$

Per dimostrare quanto detto lo statistico inglese K. Pearson lanciò una monetina 12.000 volte ottenendo 6019 teste, con frequenza relativa 0,5016; indi in una seconda serie di 12.000 lanci ottenne 6.006 teste, con frequenza relativa 0,5005.

Al giorno d'oggi, con l'avvento dei calcolatori, il signor Pearson avrebbe senza alcun dubbio scritto un bel programmino per simulare il lancio della famigerata monetina.

Sulle sue tracce abbiamo intrapreso un'analoga esperienza: abbiamo simulato con l'ausilio di un Apple II una serie di lanci di 6 dadi calcolando le frequenze relative dei risultati.

Chiunque sia dedito al gioco dei dadi sa che lanciando 2 dadi il risultato più probabile è il 7 poichè è ottenibile come:

$$1+6 \quad 2+5 \quad 3+4 \quad 4+3 \quad 5+2 \quad 6+1$$

mentre, per esempio 2 è ottenibile come:

$$1+1$$

Analogamente, lanciando 6 dadi, il risultato più probabile sarà 21 poichè è ottenibile dal massimo numero di combinazioni, ovvero 4.332 e la sua probabilità sarà:

$$P(21) = N/M = 4.332/46.656$$

dove il numero totale di combinazioni possibili ottenibili lanciando 6 dadi a 6 facce è dato dal numero di disposizioni con ripetizione di 6 elementi di classe 6, ovvero:

$$6\hat{\ } 6 = 46.656$$

Ad esempio il numero 8 si può ottenere da 21 combinazioni diverse

1	1	1	1	1	3
1	1	1	1	2	2
1	1	1	1	3	1
1	1	1	2	1	2
1	1	1	2	2	1
1	1	1	3	1	1
1	1	2	1	1	2
1	1	2	1	2	1
1	1	2	2	1	1
1	1	3	1	1	1
1	2	1	1	1	2
1	2	1	2	1	1
1	2	2	1	1	1
1	3	1	1	1	1
2	1	1	1	1	2
2	1	1	1	2	1
2	1	1	2	1	1
2	1	2	1	1	1
2	2	1	1	1	1
3	1	1	1	1	1

Figura 1. *Combinazioni possibili per ottenere 8 tirando sei dadi.*

N.	Probabilità teorica	n. Combinazioni
6	2E-05	1
7	1,2E-04	6
8	4,5E-04	21
9	1,2E-03	56
10	2,7E-03	126
11	5,4E-03	252
12	9,77E-03	456
13	0,0162	756
14	0,02488	1161
15	0,0357	1666
16	0,04816	2247
17	0.06121	2856
18	0,07353	3431
19	0,08371	3906
20	0,09047	4221
21	0,09284	4332
22	0,09047	4221
23	0,08371	3906
24	0,07353	3431
25	0,06121	2856
26	0,04816	2247
27	0,0357	1666
28	0,02488	1161
29	0,0162	756
30	9,77E-03	456
31	5,4E-03	252
32	2,7E-03	126
33	1,2E-03	56
34	4,5E-04	21
35	1,2E-04	6
36	2E-05	1

Figura 2. *Tabella delle probabilità teoriche di ciascun risultato ottenibile lanciando 6 dadi.*

come si può vedere dalla figura 1.

Il programma del listato 1, mediante 6 loop nidificati, calcola il numero di combinazioni da cui si può ottenere ciascun risultato e la sua probabilità come rapporto tra tale numero e 46.656 (cioè 6^6) che è il numero totale di combinazioni possibili; la tabulazione è in figura 2.

Queste sono le previsioni teoriche, ma, in pratica, le frequenze relative si discosteranno da esse di una certa quantità dipendente dal numero di prove effettuate.

Notiamo che la forma dell'istogramma delle probabilità in figura 3 ricorda immediatamente la curva "a campana" tipica della distribuzione di Gauss, cioè quella della distribuzione degli errori accidentali, intorno al valore vero, nella misurazione di una grandezza fisica.

Il nostro caso è invece quello della distribuzione dei risultati dei lanci intorno al valore più probabile: il 21.

Il programma del listato 2 permette di visualizzare la costruzione dell'istogramma dei risultati nel corso stesso della simulazione dei lanci.

Il lancio di un dado viene simulato per mezzo della funzione BASIC che genera dei numeri pseudocasuali: la function di libreria RND.

Essa genera numeri distribuiti con legge uniforme tra 0 ed 1; tali numeri non sono effettivamente casuali, ma vengono calcolati mediante un preciso algoritmo ed è per tale ragione che vengono definiti pseudo-casuali.

La function R(X) nella riga 130 simula il risultato di un lancio di un dado, generando un numero intero compreso tra 1 e 6.

Infatti si può dimostrare che disponendo di una serie di numeri x che seguono una legge uniforme tra 0 ed 1 se ne può ottenere una di numeri y che seguono una legge uniforme in

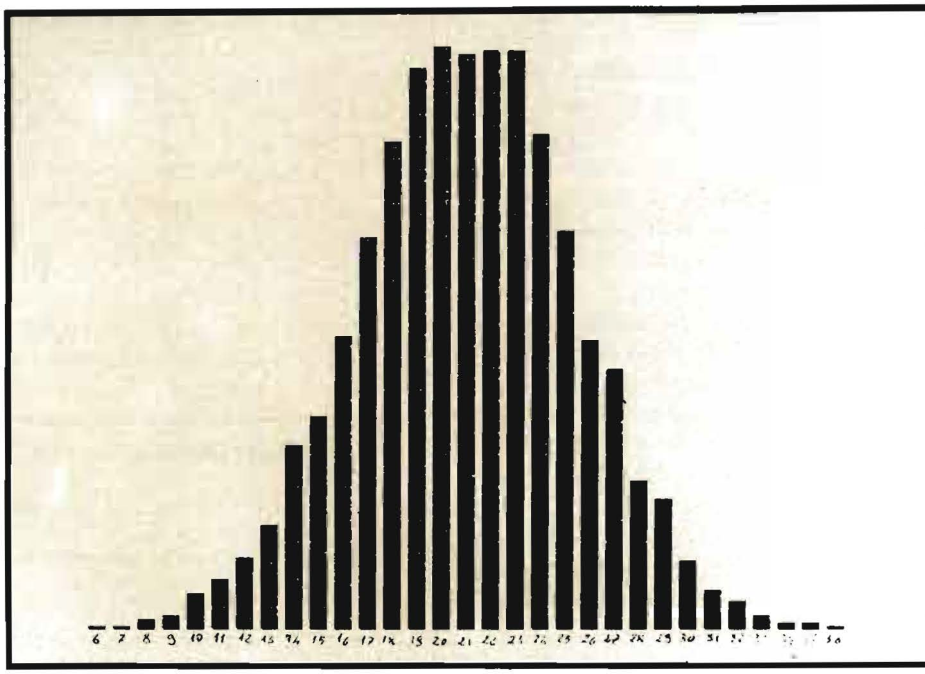


Figura 3. *Risultato dell'elaborazione con 5.000 iterazioni.*

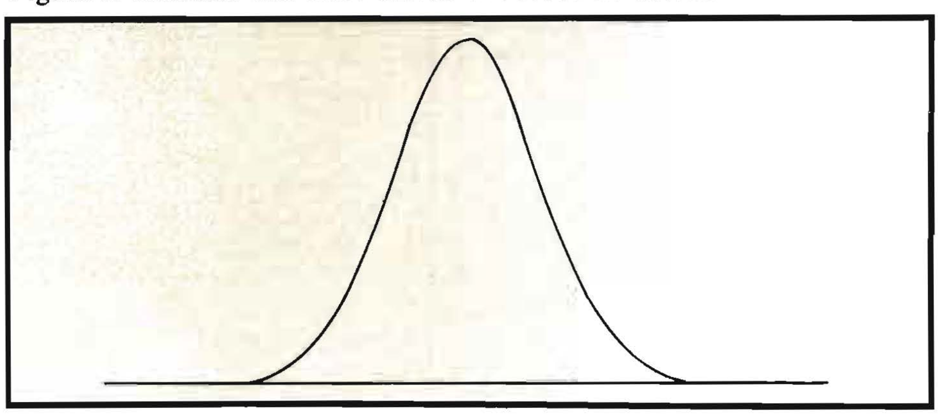


Figura 4. *Forma d'onda caratteristica della "gaussiana".*

un intervallo di estremi a e b mediante la seguente relazione:

$$y = a + x (b - a)$$

È significativo notare che l'argomento X della function R(X) è fittizio, cioè esso non compare nell'espressione il cui valore viene assegnato ad R(X); ciò è dovuto al fatto che la function di libreria RND viene aggiornata automaticamente dalla macchina e dal fatto che non vengono accettati nomi di function privi di argomento.

Il risultato del lancio di sei dadi è dato dalla somma dei risultati di sei chiamate successive della R(X) che si possono considerare indipendenti e non è ottenibile, come si potrebbe erroneamente pensare, dal calcolo diretto di un unico intero compreso tra 6 e 36.

Nella riga 220 vengono dimensionati i vettori contenenti rispettivamente i risultati dei lanci, le frequen-

ze relative e le probabilità.

Il corpo principale del programma va dalla riga 300 alla riga 520; nella riga 370 viene calcolato il risultato del lancio di 6 dadi e nella 420 viene incrementato il corrispondente contatore, quindi dalla riga 470 alla 530 vengono aggiornati il contatore dei lanci e l'istogramma.

Per evitare che l'istogramma, raggiunti certi valori, uscendo fuori dallo schermo causi un errore, è stato inserito un test nella riga 510.

Se il test risulta positivo, viene chiamata la subroutine di cambio scala; tale sottoprogramma dimezza la scala dell'istogramma e ridisegna il grafico precedente nella nuova scala, restituendo quindi il controllo al loop principale.

Si passa infine alla tabulazione dei risultati dei lanci nonchè a quella del confronto tra le probabilità calcolate teoricamente e le frequenze relative ottenute dalla simulazione della situazione reale.

A questo punto, balza all'occhio, che all'aumentare del numero di ripetizioni la forma dell'istogramma tende sempre più a quella di una "campana"; ciò farebbe sospettare che l'istogramma converga ad una gaussiana, sospetto giustificato pienamente dal sussistere delle ipotesi del teorema di De Moivre - Laplace.

Notiamo, infine, che l'avere assunto come valutazione di probabilità il limite della frequenza relativa, è giustificato, nel nostro caso particolare, in base ad alcuni importanti risultati della teoria dei grandi numeri.

Omettiamo in questa sede enunciati e dimostrazioni di tali teoremi che appesantirebbero non poco la trattazione; chi fosse interessato ad ulteriori approfondimenti può consultare, tra gli altri, i seguenti testi di Luciano Daboni:

"Calcolo delle probabilità" - ed. Boringhieri

e per una trattazione più approfondita,

"Calcolo delle probabilità ed elementi di statistica" ed. UTET. ■

```
70  DIM R%(36),P(36)
80  REM  ===============
90  REM   LOOP DI CALCOLO
100  REM  ===============
110  FOR I = 1 TO 6
120  FOR J = 1 TO 6
130  FOR K = 1 TO 6
140  FOR L = 1 TO 6
150  FOR M = 1 TO 6
160  FOR N = 1 TO 6
170 Z = I + J + K + L + M + N
180 R%(Z) = R%(Z) + 1
190  NEXT : NEXT : NEXT : NEXT : NEXT : NEXT
200  REM  ================
210  REM   LA VARIABILE C
220  REM   CONTIENE TUTTI
230  REM   I CASI POSSIBILI
240  REM  ================
250 G = 46656
260  FOR T = 6 TO 36
270 P(I) = R%(I) / G
280  REM  ==================
290  REM   APPROSSIMAZIONE
300  REM    A 5 CIFRE DECIMALI
310  REM  ==================
320 P(I) =  INT (P(I) * 100000) / 100000
330  REM  ==================
340  REM   TABULAZIONE VALORI
350  REM   DELLA PROBABILITA'
360  REM  ==================
370  PRINT I,P(I)
380  NEXT
390  END
```

Listato 1. *Programma per il calcolo delle probabilità teoriche.*

Listato 2. *Programma per la visualizzazione dell'istogramma dei risultati ottenuti lanciando sei dadi.*

```
10  TEXT : HOME : PRINT : PRINT
20  PRINT "    DISEGNO DI UN ISTOGRAMMA"
30  PRINT : PRINT : PRINT "QUESTA E' LA SIMULAZIONE"
40  PRINT "DI UNA SERIE DI LANCI DI"
50  PRINT "6 DADI CON IL TRACCIAMENTO"
60  PRINT "DELL'ISTOGRAMMA DEI RISULTATI"
70  PRINT : PRINT
80  INPUT "QUANTI LANCI DEVO FARE ? ";N
90  REM  =================
100  REM  FUNCTION CHE SIMULA
110  REM  IL LANCIO DI 1 DADO
120  REM  =================
130  DEF  FN R(X) = 1 +  INT (6 *  RND (N1))
140  VTAB 21: PRINT "          1111111111222222222233333333"
150  VTAB 22: PRINT "      6789012345678901234567890123456"
160  REM  ====================
170  REM  DIMENSIONAMENTO
180  REM  VETTORE RISULTATI
190  REM  E VETTORI PROBABILITA'
200  REM  E FREQUENZA RELATIVA
210  REM  ====================
220  DIM IST%(36),F(36),F$(36)
230  VTAB 23
240  PRINT
250  FLASH
260  REM  ===============
270  REM  FATTORE DI SCALA
280  REM  INIZIALE
290  REM  ===============
300 SCALA = 4
310  VTAB 22: PRINT "LANCI"
320  HGR : HCOLOR= 3
330  FOR I = 1 TO N:J = 0
340  REM  ===============
350  REM  LANCIO DI 6 DADI
360  REM  ===============
370 J =  FN R(X) +  FN R(X) +  FN R(X) +  FN R(X) +  FN R(X) +  FN R(X)
380  REM  =================
390  REM  INCREMENTO CONTATORE
400  REM  RISULTATI
410  REM  =================
420 IST%(J) = IST%(J) + 1
430  REM  ==========
440  REM  TRACCIAMENTO
450  REM  ISTOGRAMMA
460  REM  ==========
470 W = 7 * J
480 S = 159 - IST%(J) * SCALA
490  FOR H = W TO W + 4
500  HPLOT H,159 TO H,S
510  IF S <  = 5 THEN  GOSUB 1030
520  NEXT
530  VTAB 21: HTAB 2: PRINT I: NEXT
```

Probabilità e frequenza relativa

Seguito listato 2.

```
540  NORMAL
550  GET A$
560  REM  ================
570  REM  STAMPA RISULTATI
580  REM  ================
590  TEXT : HOME
600  PRINT : PRINT
610  PRINT "I RISULTATI SONO I SEGUENTI :"
620  PRINT : PRINT : PRINT
630  FOR W = 6 TO 9
640  PRINT " ";W;"  E' USCITO ";IST%(W); TAB( 20)"
     VOLTE"
650  NEXT
660  FOR W = 10 TO 20
670  PRINT W;"  E' USCITO ";IST%(W); TAB( 20)" VOLTE'
680  NEXT
690  GET A$: HOME : PRINT : PRINT
700  PRINT : PRINT
710  FOR W = 21 TO 36
720  PRINT W;"  E' USCITO ";IST%(W); TAB( 20)" VOLTE"
730  NEXT : PRINT : PRINT
740  GET A$: HOME
750  PRINT : PRINT
760  PRINT "TABELLA PER IL CONFRONTO": PRINT
770  PRINT "TRA FREQUENZE RELATIVE E": PRINT
780  PRINT "PROBABILITA'": PRINT
790  PRINT " N        FREQ. REL.          PROB."
800  PRINT : PRINT
810 AP = 100000
820  FOR I = 6 TO 18
830  READ P$(I)
840 F(I) = IST%(I) / N
850 F(I) =  INT (F(I) * AP) / AP
860  PRINT I; TAB( 14);F(I); TAB( 28);P$(I)
870  NEXT : GET A$: HOME
880  PRINT : PRINT
890  PRINT " N        FREQ. REL.          PROB."
900  PRINT : PRINT
910  FOR I = 19 TO 36
920  READ P$(I)
930 F(I) = IST%(I) / N
940 F(I) =  INT (F(I) * AP) / AP
950  PRINT I; TAB( 14);F(I); TAB( 28);P$(I)
960  NEXT
970  DATA  .00002,.00012,.00045,.00200,.00270,.00540,.00977,.01620,.02488,
     .03570,.04816,.06121,.07353,.08371,.09047,.09284,.09047,.08371,.07353
     ,.06121,.04816,.03570,.02488,.01620,.00977,.00540,.00270,.00200,.0004
     5,.00012,.00002
980  GET A$: HOME : END
990  REM  ================
1000  REM  SUBROUTINE CAMBIO
1010  REM  DI SCALA
1020  REM  ================
1030  HGR : HCOLOR= 3
1040 SCALA = SCALA / 2
1050  FOR W = 42 TO 252 STEP 7
1060 K = W / 7
1070 S = 159 - IST%(K) * SCALA
1080  FOR H = W TO W + 4
1090  HPLOT H,159 TO H,S
1100  NEXT : NEXT
1110  RETURN
```

Programmer's tool kit per Commodore 64

- seconda parte -

Il programma in linguaggio macchina

di *Alessandro Guida*

In questa seconda parte viene presentato il listato disassemblato del programma proposto nel numero 12/13 di **Personal Software**.

I numerosi commenti inseriti dovrebbero permettere una buona comprensione delle varie routine definite.

Vediamo comunque qualche spiegazione aggiuntiva del programma.

Una delle cose più interessanti è vedere come sia possibile aggiungere nuovi comandi al BASIC del C64.

L'estensione del BASIC può essere fatta a due livelli.

Si può modificare la Tokenisation routine cambiando il vettore che ne conserva l'indirizzo di partenza ($0304). Ma per comprendere questo tipo di intervento bisogna sapere come funziona l'interprete BASIC.

Quando introduciamo una linea da tastiera (sia di programma sia da eseguire direttamente) questa viene "tokenizzata", ossia le parole chiave, in essa contenute, vengono trasformate in codici di un solo byte. In ogni buon manuale sui sistemi Commodore è riportato un elenco di questi codici chiamati, appunto, TOKEN. Durante l'esecuzione della linea, quando viene incontrato un token, lo stesso token è utilizzato come puntatore per leggere in una apposita tavola l'indirizzo della routine che esegue quel particolare comando. La routine che svolge questa operazione è la Command Dispatcher Routine (puntata dal vettore $0308).

Come si può intuire questo metodo è tutt'altro che semplice, poichè è necessario scrivere una nuova Tokenisation routine, una nuova Command Disp. routine e una nuova tavola di indirizzi.

Meno complesso è, invece, il sistema utilizzato in questo articolo.

Siccome tutti i comandi che vogliamo implementare verranno utilizzati solo in modo diretto, e non inseriti in programmi, modificheremo la Handle Basic Line routine (è la routine che gestisce le line digitate da tastiera).

Quando il vostro 64 è in attesa di comandi, ad esempio subito dopo l'accensione, sta eseguendo proprio questa routine. In particolare è occupato a gestire l'input da tastiera (e il conseguente eco sullo schermo) e vi rimane finchè non viene premuto il tasto di RETURN.

A questo punto legge la linea dello schermo sulla quale giace il cursore e la riporta in un buffer (BASIC input buffer) che parte da $0200. Subito dopo controlla se è una linea da eseguire direttamente, o è una linea di programma perchè comincia con un numero. Nel secondo caso, se c'è già nel programma una linea con lo stesso numero la cancella, poi tokenizza la nuova linea nel buffer, apre lo spazio necessario in memoria e la inserisce. Se invece è una linea da eseguire in modo diretto, la tokenizza e passa subito alla Command Disp. routine che la esegue.

L'indirizzo di partenza della Handle BASIC Line routine è conservato nel vettore $0302, che modificheremo introducendo l'indirizzo della nostra nuova routine.

Intercettazione dei nuovi comandi

La nuova routine è nel disassemblato a partire da $9A50. Questa controlla se la prima parola nel BASIC input buffer è una di quelle conservate nella tavola che va da $9900 a $993C. Se non lo è torna alla routine di Handle originale ($A48A), in caso contrario carica l'indirizzo della routine che esegue il comando in questione e passa a eseguirla.

Routine di Merge

Da $993D a $9963 la routine riceve i parametri del programma che vogliamo includere in quello già in memoria (nome file e periferica dove leggerlo).

Da $9965 a $9978 setta i parametri del file da aprire assegnando anche un comando secondario "0". Questo comando serve a fare in modo che nella operazione di load seguente sia possibile rilocare il programma in una zona qualsiasi della memoria ignorando l'header della registrazione.

Da $997B a $998C carica il programma a partire da 256 byte dopo il programma principale, segnalando eventuali errori in fase di lettura.

A questo punto il resto della routine si occuperà di leggere le linee del programma una per una, memorizzare la linea letta nel BASIC input buffer, cambiando i token in parole chiave, e chiamare quella parte della Handle BASIC routine che si occupa di inserire le nuove linee BASIC nei programmi.

Poiché, però, la routine di inserimento termina con un JMP ($0302) dovremo modificare ancora questo vettore se vogliamo che dopo l'inserimento della linea il controllo torni alla routine di Merge.

Alla fine dell'intera operazione riporteremo il valore originale in $0302.

Routine di Key

La parte principale della routine che abilita l'uso dei tasti di funzione è la nuova IRQ routine. Ricordo che questa è una routine chiamata ogni sessantesimo di secondo da un timer hardware, e che si occupa tra l'altro della gestione della tastiera.

Cambiando il vettore che ne contiene l'indirizzo possiamo aggiungere una nostra routine (da $9A90 a $9AD6) che controlla se è stato premuto qualche tasto di funzione. Se ciò è accaduto mette nel buffer la stringa associata al tasto altrimenti prosegue con la routine di IRQ normale.

Il resto della funzione KEY è svolto dalla routine in $9AD8. Le stringhe vengono memorizzate a partire da $9F80 e per ognuna sono riservati 16 byte (la stringa termina al primo byte uguale zero). Di conseguenza per conoscere il punto dove una stringa va memorizzata o letta basta moltiplicare il numero del tasto funzione per 16, ossia bisogna eseguire 4 ASL (Shift di 1 bit a sinistra).

Routine di Dump

La routine parte da $9B55. Legge tutte le variabili presenti in memoria partendo dalla fine del BASIC (puntatore $2D, 2E) sino alla fine della zona delle variabili (puntatore $2F, 30). Le variabili occupano tutte 7 byte ciascuna, indipendentemente dal tipo. I primi due byte sono del nome. Se la variabile è del tipo intero sia il primo che il secondo carattere del nome hanno il bit 7 settato (OR 80). Se è una variabile stringa solo il secondo carattere, e se è del tipo floating point nessuno dei due è alterato.

In $9BE4 viene chiamata la routine $BBA2 che carica un numero in floating point dalla variabile nell'accumulatore 1. Questo perchè il formato è leggermente diverso. Infatti nelle variabili float il bit più significativo rappresenta il segno mentre nell'accumulatore è la locazione $66 ad indicare il segno:
(00 = + FF = –).

Routine di Find

È da notare in $9C2D la chiamata alla Tokenisation routine. Così è possibile rintracciare anche i comandi BASIC all'interno del programma. Questo, però costringe a racchiudere le stringhe tra virgolette, altrimenti una parola come FORSE verrebbe interpretata come lo statement FOR seguito dalla variabile SE.

La linea tokenizzata risiede nel BASIC Input Buffer.

Routine di Help

Per implementare l'Help è stato necessario modificare due routine dell'interprete BASIC: l'Error Routine e la Print Token Routine. Come al solito cambieremo i vettori che puntano a queste routine, nell'ordine $0300 e $0306.

La nuova Error Routine è solo una aggiunta che permette di memorizzare il numero di linea in cui si è verificato l'errore (contenuto in $39,3A), e l'indirizzo (letto in $7A, 7B) in apposite locazioni che verranno poi richiamate dalla routine di Help vera e propria. Fatto questo si torna alla routine normale per la gestione tradizionale dei messaggi d'errore.

La print Token Routine è, invece, una routine che viene chiamata per ogni carattere in fase di List di una linea di programma BASIC e che decide se si tratta di un carattere ASCII normale o di un token, nel qual caso stampa la parola chiave corrispondente. La modifica consiste nel decrementare, ogni volta che viene chiamata, il registro $9A4E che contiene il numero del carattere, a partire dall'inizio della linea, responsabile dell'errore. Giunto a zero tale contatore, e quindi raggiunto il punto dell'errore, provvede a stampare un punto interrogativo.

Inoltre, poichè la routine di List termina con un salto alla Error Routine (ma solo per la stampa del READY), nella nuova Error Routine è incluso anche un controllo sulla locazione $9A4F che indica se la routine di List era stata chiamata dalla funzione di Help, e quindi se terminarla con un RTS.

A questo punto la routine di Help si riduce solo al calcolo della distanza dall'inizio della linea dell'errore e ad una chiamata della routine di List.

La routine di List, in generale, stampa tutte le linee del programma partendo da quella all'indirizzo contenuto in $5F,60 e terminando a quella di numero uguale al contenuto di $14,15. Se, quindi, $5F,60 e $14,15 si riferiscono alla stessa linea ne verrà stampata una sola.
L'indirizzo di partenza della linea viene cercato dalla routine $A613, e conservato in $5F,60.

Routine Input due Parametri

È una routine, che parte da $9D0E, comune a diversi comandi che richiedono l'input di due para-

Programmer's tool kit per Commodore 64

metri.
Effettua il controllo della sintassi e memorizza i due valori in $9A44, $9A45 e $9A46, 9A47.

Routine Auto

Come abbiamo già detto le linee digitate da tastiera vengono gestite dalla Handle BASIC Line Routine. Quindi l'attivazione della funzione Auto consiste solo nell'aggiunta alla Handle B. L. routine di una parte che provvede a calcolare il nuovo numero di linea, stamparlo sullo schermo e quindi ritornare il controllo alla routine originale.

Routine Trace

Questa funzione interviene sulla routine, di cui abbiamo già parlato, Command Disp. (o Baslink Statement) che lancia l'esecuzione di ogni istruzione del programma. Poichè, però, ci possono essere più comandi sulla stessa linea, è necessario controllare se è cambiato il numero di linea prima di stamparlo. La funzione di Step, se inserita, blocca l'esecuzione finchè non viene premuto un tasto qualsiasi. Il controllo è effettuato testando la locazione $C5 che contiene il codice del tasto premuto (40 se nessuno).

Routine Delete

La routine di Delete, dopo aver letto i valori dei due parametri attraverso la Routine Input 2 Parametri, cancella le linee BASIC utilizzando la stessa parte della Handle BASIC Line Routine già utilizzata dalla routine di Merge. In questo caso, però, il buffer è lasciato vuoto per permettere la sola cancellazione delle linee. ■

Questo programma è disponibile su cassetta: vedere coupon a pagina 78.

Figura 1. *Il listato del programma.*

```
033C                          ;*PROGRAMMER'S TOOL KIT*
033C                          ;
033C                          ;AMPEC div. Software
033C                          ;   via Panzani,13
033C                          ; 50123  FIRENZE
033C                          ;
033C                          ; dir. resp. Alessandro Guida
033C                          ;
033C                          ;---------------------------
033C                          ;ROUTINE DI INIZIALIZZAZIONE
033C   78         SEI         ;---------------------------
033D   A9 9F      LDA 9F      ;
033F   85 FC      STA $FC     ;Puntatore zona stringhe
0341   A9 80      LDA 80      ;istruzione KEY ($9F80)
0343   85 FB      STA $FB     ;---------------------------
0345   A9 50      LDA 50      ;Aggiorna vettore
0347   8D 02 03   STA $0302   ;Command Handler Routine
034A   A9 9A      LDA 9A      ;   ($9A50)
034C   8D 03 03   STA $0303   ;---------------------------
034F   A9 00      LDA 00      ;
0351   A0 7F      LDY 7F      ;Azzera zona di memoria
0353   91 FB      STA ($FB),Y ;dedicata alle stringhe
0355   88         DEY         ;istruzione KEY
0356   10 FB      BPL $0353   ;---------------------------
0358   A9 9A      LDA 9A      ;
035A   8D 15 03   STA $0315   ;Aggiorna vettore IRQ
035D   A9 90      LDA 90      ;   ($9A90)
035F   8D 14 03   STA $0314   ;---------------------------
0361   A9 8F      LDA 8F      ;
0363   8D 00 03   STA $0300   ;Aggiorna vettore
0366   A9 9C      LDA 9C      ;Error Routine ($9C8F)
0368   8D 01 03   STA $0301   ;
036B   58         CLI         ;
036C   60         RTS         ;
9900                          ;Nuovo TOP MEM ($37,38)
9900                          ;PAROLE CHIAVE E INDIRIZZI
9900                          ;DELLE ROUTINE RELATIVE
9900                          ;---------------------------
9900   48 45 4C 50            ;BYT:'HELP
9904   9C C6                  ;---------------------------
9906   41 55 54 4F            ;BYT:'AUTO
990A   9D 61                  ;---------------------------
990C   46 49 4E 44            ;BYT:'FIND
9910   9C 12                  ;---------------------------
9912   44 55 4D 50            ;BYT:'DUMP
9916   9B 56                  ;---------------------------
9918   52 45 4E               ;BYT:'REN
991B   9F 12                  ;---------------------------
991D   54 52 4F               ;BYT:'TRO
9920   9D C0                  ;---------------------------
9922   4D 45 52 47 45         ;BYT:'MERGE
9927   99 3D                  ;---------------------------
9929   10 10 10 10            ;spazio libero
992D   A0 A0                  ;---------------------------
992F   44 45 4C 45 54 45      ;BYT:'DELETE
9935   9E AF                  ;---------------------------
9937   4B 45 59               ;BYT:'KEY
993A   9A D8                  ;
993C   00                     ;
993D   68         PLA         ;ROUTINE DI MERGE
993E   68         PLA         ;sntx. MERGE("file")(,dev)
993F   A2 00      LDX 00      ;---------------------------
9941   20 73 00   JSR $0073   ;
9944   F0 1F      BEQ $9965   ;Controlla la sintassi del
9946   C9 22      CMP 22      ;comando e in caso di errore
9948   F0 07      BEQ $9951   ;salta a $AF08 (stampa di
994A   C9 2C      CMP 2C      ;SINTAX ERROR e READY).
994C   F0 12      BEQ $9960   ;
994E   4C 08 AF   JMP $AF08   ;Conserva il filename nel
9951   20 73 00   JSR $0073   ;buffer BASIC ($0200,0258)
9954   F0 F8      BEQ $994E   ;e il numero della periferica
9956   C9 22      CMP 22      ;in $FE.
9958   F0 E7      BEQ $9941   ;Come default si ha il
995A   9D 00 02   STA $0200,X ;filename = "", e il
995D   E8         INX         ;dev.=1 (registratore).
995E   D0 F1      BNE $9951   ;
9960   20 73 00   JSR $0073   ;
9963   B0 E9      BCS $994E   ;---------------------------
9965   85 FE      STA $FE     ;
9967   8A         TXA         ;X=len("filename")
9968   A2 00      LDX 00      ;Attribuisce al file da
996A   A0 02      LDY 02      ;aprire il nome conservato
996C   20 BD FF   JSR $FFBD   ;nel buffer ($0200).
996F   A6 FE      LDX $FE     ;---------------------------
9971   D0 01      BNE $9974   ;Assegna al file il numero 1,
9973   E8         INX         ;il num. della periferica e
9974   A0 00      LDY 00      ;il comando 0 (serve a igno-
9976   A9 01      LDA 01      ;rare l'header della regist.)
9978   20 BA FF   JSR $FFBA   ;
997B   A6 2D      LDX $2D     ;---------------------------
997D   86 F7      STX $F7     ;
997F   A4 2E      LDY $2E     ;Carica il file gia' definito
9981   C8         INY         ;in memoria, subito dopo la
9982   84 F8      STY $F8     ;fine del programma, partendo
9984   A9 00      LDA 00      ;dall'indirizzo in($F7,F8)
9986   20 D5 FF   JSR $FFD5   ;
9989   90 04      BCC $998F   :Se durante la fase di LOAD
998B   AA         TAX         ;ci sono errori: C=1;allora
998C   6C 00 03   JMP ($0300) ;stampa un messaggio d'errore
998F   86 F9      STX $F9     ;---------------------------
9991   84 FA      STY $FA     ;
9993   A9 9D      LDA 9D      ;Aggiorna il vettore
9995   8D 02 03   STA $0302   ;Command Handler Routine
9998   A9 99      LDA 99      ;       ($999D)
999A   8D 02 03   STA $0303   ;---------------------------
999D   A0 01      LDY 01      ;ROUTINE DI INSERIMENTO LINEE
999F   84 0F      STY $0F     ;'flag delle virgolette
99A1   B1 F7      LDA ($F7),Y ;Se il HB del link della
99A3   D0 03      BNE $99A8   ;linea puntata da ($F7,F8) e'
99A5   4C 05 9F   JMP $9F05   ;=0 allora passa a restaurare
99A5                          ;il vettore ($0302) e termina
99A8   C8         INY         ;---------------------------
99A9   B1 F7      LDA ($F7),Y ;
99AB   85 14      STA $14     ;carica il numero della linea
99AD   C8         INY         ;in $14,$15
99AE   B1 F7      STA $15     ;
99B0   C8         INY         ;
99B1   84 FE      STY $FE     ;---------------------------
99B3   A4 14      LDY $14     ;
99B5   A5 15      LDA $15     ;Converte il numero di linea
99B7   20 91 B3   JSR $B391   ;da fix. in float.point e
99BA   20 DD BD   JSR $BDDD   ;da float.point in ASCII
99BD   A2 00      LDX 00      ;---------------------------
99BF   BD 01 01   LDA $0101,X ;
99C2   F0 06      BEQ $99CA   ;ricopia il numero dal buffer
99C4   9D 00 02   STA $0200,X ;ASCII al BASIC Input Buffer
99C7   E8         INX         ;
99C8   D0 F5      BNE $99BF   ;---------------------------
99CA   A4 FE      LDY $FE     ;
99CC   D0 0A      BNE $99D8   ;
99CE   C9 22      CMP 22      ;
```

Programmer's tool kit per Commodore 64

Seguito figura 1.

```
99D0   D0 06      BNE $99D8     ;
99D2   A5 0F      LDA $0F       ;
99D4   49 FF      EOR FF        ;
99D6   85 0F      STA $0F       ;Finisce di copiare la linea
99D8   B1 F7      LDA ($F7),Y   ;dalla memoria al Basic Input
99DA   F0 3F      BEQ $9A1B     ;Buffer, convertendo i token
99DC   30 07      BMI $99E5     ;in parole chiave.
99DE   9D 00 02   STA $0200,X   ;'Non e' un token
99E1   C8         INY           ;
99E2   E8         INX           ;
99E3   D0 E9      BNE $99CE     ;'Prossimo carattere
99E5   C9 FF      CMP FF        ;
99E7   F0 F5      BEQ $99DE     ;'Non e' un token ma PI
99E9   24 0F      BIT $0F       ;
99EB   30 F1      BMI $99DE     ;'E' un carattere grafico
99ED                            ;---------------------------
99ED   38         SEC           ;ROUTINE DI CONVERSIONE DEI
99EE   E9 7F      SBC 7F        ;TOKEN IN KEYWORD.
99F0   86 49      STX $49       ;
99F2   84 FE      STY $FE       ;
99F4   AA         TAX           ;
99F5   A0 FF      LDY FF        ;
99F7   CA         DEX           ;
99F8   F0 08      BEQ $9A02     ;
99FA   C8         INY           ;
99FB   B9 9E A0   LDA $A09E,Y   ;
99FE   10 FA      BPL $99FA     ;
9A00   30 F5      BMI $99F7     ;
9A02   A6 49      LDX $49       ;
9A04   C8         INY           ;
9A05   B9 9E A0   LDA $A09E,Y   ;
9A08   30 06      BMI $9A10     ;
9A0A   9D 00 02   STA $0200,X   ;
9A0D   E8         INX           ;
9A0E   D0 F4      BNE $9A04     ;
9A10   29 7F      AND 7F        ;
9A12   9D 00 02   STA $0200,X   ;
9A15   A4 FE      LDY $FE       ;
9A17   C8         INY           ;
9A18   E8         INX           ;
9A19   D0 BD      BNE $99D8     ;
9A1B   9D 00 02   STA $0200,X   ;---------------------------
9A1E   98         TYA           ;
9A1F   38         SEC           ;Calcola l'indirizzo della
9A20   65 F7      ADC $F7       ;prossima linea del file e
9A22   85 F7      STA $F7       ;aggiorna $F7,$F8
9A24   90 02      BCC $9A28     ;
9A26   E6 F8      INC $F8       ;
9A28   A2 FF      LDX FF        ;
9A2A   A0 01      LDY 01        ;
9A2C   4C 86 A4   JMP $A486     ;Salta alla routine di HANDLE
9A2E   00 00 00                 ;NEW BASIC LINE,che termina
9A31   00 00 00                 ;con un JMP($0302)
9A34   00 00 00                 ;---------------------------
9A37                            ;ZONA DI MEMORIA A DISPOSI-
9A37                            ;ZIONE DELLE ROUTINE DEL
9A37                            ;PROGRAMMER'S TOOL KIT
9A37                            ;
9A37   4B 45 59                 ;BYT:'KEY
9A3A   00 00 00                 ;
9A3D   4C 00 00   JMP $0000     ;Vettore di salto per le rou-
9A3D                            ;tine dei nuovi comandi.
9A40   00 00                    ;(HELP) numero linea
9A42   00 00                    ;(HELP) puntatore Basic
9A44   00 00                    ;Parametro 'a'
9A46   00 00                    ;Parametro 'b'
9A48   00                       ;
9A49   00                       ;(STEP) 0=no 1=si
9A4A   00 00                    ;(STEP) ultima linea eseg.
9A4C   00 00                    ;
9A4E   00                       ;(HELP)
9A4F   00                       ;(ERROR) FF=List 00=Ready
9A50                            ;NUOVA HANDLE BASIC LINE ROUT
9A50                            ;---------------------------
9A50   20 60 A5   JSR $A560     ;'riceve linea dalla tastiera
9A53   86 7A      STX $7A       ;
9A55   84 7B      STY $7B       ;'aggiorna puntatore CHARGET
9A57   A0 00      LDY 00        ;
9A59   A2 00      LDX 00        ;
9A5B   B1 37      LDA ($37),Y   ;'prende carattere dall'elen-
9A5B                            ;'co nuovi comandi
9A5D   D0 03      BNE $9A62     ;'se = 0 l'elenco e' finito
9A5F   4C 8A A4   JMP $A48A     ;'va alla routine normale
9A62   30 12      BMI $9A76     ;'parola trovata in elenco
9A64   DD 00 02   CMP $0200,X   ;'confronta con car.in buffer
9A67   D0 04      BNE $9A6D     ;
9A69   C8         INY           ;
9A6A   E8         INX           ;
9A6B   D0 EE      BNE $9A5B     ;'prossimo carattere
9A6D   C8         INY           ;---------------------------
9A6E   B1 37      LDA ($37),Y   ;cerca la prossima parola
9A70   10 FB      BPL $9A6D     ;chiave in elenco
9A72   C8         INY           ;
9A73   C8         INY           ;
9A74   D0 E3      BNE $9A59     ;---------------------------
9A76   8D 3F 9A   STA $9A3F     ;mette l'indirizzo della
9A79   C8         INY           ;routine interessata nel
9A7A   B1 37      LDA ($37),Y   ;vettore di salto
9A7C   8D 3E 9A   STA $9A3E     ;
```

Seguito listato 1.

```
9A7F   18         CLC           ;---------------------------
9A80   A9 FF      LDA FF        ;indica che e' una linea
9A82   85 3A      STA $3A       ;eseguita in modo diretto
9A84   CA         DEX           ;---------------------------
9A85   86 7A      STX $7A       ;riaggiorna puntatore
9A87   E6 7B      INC $7B       ;  CHARGET
9A89   20 3D 9A   JSR $9A3D     ;'esegue la routine del
9A8C                            ; comando attraverso il
9A8C                            ; vettore $9A3D.
9A8C   4C 74 A4   JMP $A474     ;'stampa READY e torna in
9A8F                            ; attesa di una nuova linea.
9A8F                            ;La routine in A474 termina
9A8F ;con JMP($0302). 9A8F EA  NOP ;
9A90                            ;NUOVA IRQ ROUTINE
9A90                            ;---------------------------
9A90   A5 C5      LDA $C5       ;'legge codice tasto premuto
9A92   C5 02      CMP $02       ;---------------------------
9A94   F0 0D      BEQ $9AA3     ;controlla se e' lo stesso di
9A96   85 02      STA $02       ;prima, se si non lo testa
9A98   38         SEC           ;---------------------------
9A99   E9 03      SBC 03        ;controlla se e' un tasto di
9A9B   30 06      BMI $9AA3     ;funzione,se no torna alla
9A9D   F0 07      BEQ $9AA6     ;routine IRQ originale.
9A9F   C9 04      CMP 04        ;
9AA1   30 05      BMI $9AA8     ;
9AA3   4C 31 EA   JMP $EA31     ;'routine IRQ originale
9AA6   A9 04      LDA 04        ;---------------------------
9AA8   A8         TAY           ;
9AA9   88         DEY           ;
9AAA   98         TYA           ;
9AAB   0A         ASL           ;calcola l'offset per leggere
9AAC   0A         ASL           ;la stringa associata al
9AAD   0A         ASL           ;tasto funz. premuto.
9AAE   0A         ASL           ;
9AAF   0A         ASL           ;
9AB0   85 FD      STA $FD       ;
9AB2   AD 8D 02   LDA $028D     ;'controlla se e' premuto lo
9AB5   29 01      AND 01        ;' SHIFT
9AB7   F0 07      BEQ $9AC0     ;
9AB9   18         CLC           ;
9ABA   A5 FD      LDA $FD       ;
9ABC   69 10      ADC 10        ;
9ABE   85 FD      STA $FD       ;
9AC0   A4 FD      LDY $FD       ;---------------------------
9AC2   A2 00      LDX 00        ;legge la stringa e la mette
9AC4   B1 FB      LDA ($FB),Y   ;nel buffer della tastiera
9AC6   F0 07      BEQ $9ACF     ;
9AC8   9D 77 02   STA $0277,X   ;
9ACB   C8         INY           ;
9ACC   E8         INX           ;
9ACD   D0 F5      BNE $9AC4     ;
9ACF   86 C6      STX $C6       ;'numero caratteri nel buff.
9AD1   68         PLA           ;---------------------------
9AD2   A8         TAY           ;
9AD3   68         PLA           ;chiusura della routine
9AD4   AA         TAX           ;  di IRQ
9AD5   68         PLA           ;
9AD6   40         RTI           ;
9AD7   EA         NOP           ;
9AD8                            ;ROUTINE DI KEY
9AD8                            ;sntx: KEY(n,)(stringa)(<)
9AD8                            ;---------------------------
9AD8   20 73 00   JSR $0073     ;se non ci sono altri caratt.
9ADB   F0 3B      BEQ $9B18     ;   va alla routine di list
9ADD   90 03      BCC 9AE2      ;se il car. seguente non e'un
9ADF   4C 08 AF   JMP $AF08     ;   numero SINTAX ERROR
9AE2   85 FE      STA $FE       ;
9AE4   C6 FE      DEC $FE       ;
9AE6   A9 2C      LDA 2C        ;
9AE8   A0 01      LDY 01        ;'controlla che segua una
9AEA   20 01 AF   JSR $AF01     ;' virgola
9AED   A5 FE      LDA $FE       ;---------------------------
9AEF   0A         ASL           ;calcola l'offset relativo a
9AF0   0A         ASL           ;'n' che punta la zona dove
9AF1   0A         ASL           ;deve essere memorizzata la
9AF2   0A         ASL           ;stringa
9AF3   A8         TAY           ;---------------------------
9AF4   A2 0A      LDX 0A        ;num.max di caratteri
9AF6   20 73 00   JSR $0073     ;
9AF9   90 0F      BCC $9B0A     ;copia la stringa in memoria
9AFB   F0 18      BEQ $9B15     ;
9AFD   C9 5F      CMP 5F        ;
9AFF   F0 07      BEQ $9B08     ;
9B01   20 13 B1   JSR $B113     ;
9B04   B0 04      BCS $9B0A     ;
9B06   90 D7      BCC $9ADF     ;
9B08   A9 0D      LDA 0D        ;
9B0A   91 FB      STA ($91),Y   ;
9B0C   C8         INY           ;
9B0D   CA         DEX           ;
9B0E   D0 E6      BNE $9AF6     ;
9B10   A2 17      LDX 17        ;STRING TOO LONG ERROR
9B12   6C 00 03   JMP ($0300)   ;
9B15   91 FB      STA ($FB),Y   ;
9B17   60         RTS           ;
9B18   A2 00      LDX 00        ;---------------------------
9B1A   86 FE      STX $FE       ;routine LIST TASTI-STRINGHE
9B1C   A9 37      LDA 37        ;
9B1E   A0 9A      LDY 9A        ;
9B20   20 1E AB   JSR $AB1E     ;stampa "KEY"
9B23   A5 FE      LDA $FE       ;
9B25   18         CLC           ;
9B26   69 31      ADC 31        ;
9B28   20 D2 FF   JSR $FFD2     ;stampa 'n'
9B2B   A9 2C      LDA 2C        ;
```

PERSONAL SOFTWARE

Programmer's tool kit per Commodore 64

Seguito listato 1.

```
9B2D    20 D2 FF    JSR $FFD2     ;stampa ','
9B30    A5 FE       LDA $FE       ;
9B32    0A          ASL           ;
9B33    0A          ASL           ;
9B34    0A          ASL           ;
9B35    0A          ASL           ;
9B36    A8          TAY           ;
9B38    B1 FB       LDA ($FB),Y   ;
9B39    F0 0C       BEQ $9B47     ;
9B3B    C9 0D       CMP 0D        ;
9B3D    D0 02       BNE $9B41     ;
9B3F    A9 5F       LDA 5F        ;
9B41    20 D2 FF    JSR $FFD2     ;stampa 1 car.della stringa
9B44    C8          INY           ;
9B45    D0 F0       BNE 9B38      ;prossimo carattere
9B47    A9 0D       LDA 0D        ;
9B49    20 D2 FF    JSR $FFD2     ;
9B4C    E6 FE       INC $FE       ;
9B4E    A5 FE       LDA $FE       ;
9B50    C9 08       CMP 08        ;
9B52    D0 C8       BNE $9B1C     ;prossimo tasto
9B54    60          RTS           ;
9B55                              ;ROUTINE DI DUMP
9B55                              ;sntx : DUMP.
9B55    EA          NOP           ;---------------------------
9B56    20 73 00    JSR $0073     ;se ci sono altri caratteri
9B59    F0 03       BEQ $9B5E     ;
9B5B    4C 08 AF    JMP $AF08     ;SINTAX ERROR
9B5E                              ;---------------------------
9B5E    A5 2D       LDA $2D       ;in $FE,$FF inizio variabili
9B60    85 FE       STA $FE       ;
9B62    A5 2E       LDA $2E       ;
9B64    85 FF       STA $FF       ;
9B66    A0 00       LDY 00        ;---------------------------
9B68    A9 01       LDA 01        ;
9B6A    85 0D       STA $0D       ;$0D = tipo variabile
9B6C                              ; 01=int.,00=stringa,FF=flo.
9B6C    B1 FE       LDA ($FE),Y   ;
9B6E    85 46       STA $46       ;prima lettera variabile
9B70    30 02       BMI $9B74     ;se > 80 variab.intera
9B72    84 0D       STY $0D       ;
9B74    C8          INY           ;
9B75    B1 FE       LDA ($FE),Y   ;seconda lettera variabile
9B77    85 45       STA $45       ;
9B79    30 04       BMI $9B80     ;se > 80 var.intera o stringa
9B7B    A9 FF       LDA FF        ;
9B7D    85 0D       STA $0D       ;
9B7F    A2 02       LDX 02        ;---------------------------
9B81    B5 44       LDA $44,X     ;
9B83    29 7F       AND 7F        ;stampa nome variabile
9B85    20 D2 FF    JSR $FFD2     ;
9B88    CA          DEX           ;
9B89    D0 F6       BNE $9B81     ;---------------------------
9B8B    A5 0D       LDA $0D       ;
9B8D    30 06       BMI $9B95     ;
9B8F    18          CLC           ;stampa '%' o '$'
9B90    69 24       ADC 24        ;
9B92    20 D2 FF    JSR $FFD2     ;
9B95    A9 3D       LDA 3D        ;
9B97    20 D2 FF    JSR $FFD2     ;stampa '='
9B9A    EA          NOP           ;
9B9B    EA          NOP           ;
9B9C    A5 0D       LDA $0D       ;
9B9E    F0 4A       BEQ $9BEA     ;variabile stringa
9BA0    30 36       BMI $9BD8     ;var.floating point
9BA2                              ;---------------------------
9BA2    C8          INY           ;VARIABILE INTERA
9BA3    B1 FE       LDA ($FE),Y   ;
9BA5    48          PHA           ;
9BA6    C8          INY           ;carica HB e LB del numero
9BA7    B1 FE       LDA ($FE),Y   ;  intero in A,Y
9BA9    A8          TAY           ;
9BAA    68          PLA           ;
9BAB    20 91 B3    JSR $B391     ;converte da int. in float.
9BAE    20 DD BD    JSR $BDDD     ;conv. da float. in ASCII
9BB1    A9 00       LDA 00        ;
9BB3    A0 01       LDY 01        ;
9BB5    20 1E AB    JSR $AB1E     ;stampa i caratteri ASCII
9BB8    A9 0D       LDA 0D        ;
9BBA    20 D2 FF    JSR $FFD2     ;stampa un CR
9BBD    18          CLC           ;---------------------------
9BBE    A5 FE       LDA $FE       ;calcola l'indirizzo della
9BC0    69 07       ADC 07        ;prossima variabile
9BC2    85 FE       STA $FE       ;
9BC4    A5 FF       LDA $FF       ;
9BC6    69 00       ADC 00        ;
9BC8    85 FF       STA $FF       ;
9BCA    C5 30       CMP $30       ;controlla se ci sono altre
9BCC    F0 03       BEQ $9BD1     ;variabili
9BCE    4C 66 9B    JMP $9B66     ;prossima variabile
9BD1    A5 FE       LDA $FE       ;
9BD3    C5 2F       CMP $2F       ;
9BD5    90 F7       BCC $9BCE     ;
9BD7    60          RTS           ;
9BD8    18          CLC           ;---------------------------
9BD9    C8          INY           ;VARIABILE FLOATING
9BDA    98          TYA           ;
9BDB    65 FE       ADC $FE       ;
9BDD    AA          TAX           ;carica il numero floating
9BDE    A5 FF       LDA $FF       ;
9BE0    69 00       ADC 00        ;
```

```
9BE2    A8          TAY           ;
9BE3    8A          TXA           ;
9BE4    20 A2 BB    JSR $BBA2     ;
9BE7    4C AE 9B    JMP $9BAE     ;---------------------------
9BEA    A9 22       LDA 22        ;VARIABILE STRINGA
9BEC    20 D2 FF    JSR $FFD2     ;'stampa "
9BEF    C8          INY           ;
9BF0    B1 FE       LDA ($FE),Y   ;'legge la lung.della stringa
9BF2    F0 16       BEQ $9C0A     ;'stringa vuota
9BF4    AA          TAX           ;
9BF5    C8          INY           ;
9BF6    B1 FE       LDA ($FE),Y   ;legge indirizzo corpo strin.
9BF8    85 69       STA $69       ;
9BFA    C8          INY           ;
9BFB    B1 FE       LDA ($FE),Y   ;
9BFD    85 6A       STA $6A       ;
9BFF    A0 00       LDY 00        ;
9C01    B1 69       LDA ($69),Y   ;stampa la stringa
9C03    20 D2 FF    JSR $FFD2     ;
9C06    C8          INY           ;
9C07    CA          DEX           ;
9C08    D0 F7       BNE $9C01     ;
9C0A    A9 22       LDA 22        ;stampa "
9C0C    20 D2 FF    JSR $FFD2     ;
9C0F    4C B8 9B    JMP $9BB8     ;prossima variabile
9C12                              ;ROUTINE FIND
9C12                              ;sntx : FIND,"stringa"/testo
9C12                              ;---------------------------
9C12    20 73 00    JSR $0073     ;
9C15    C9 02       CMP 2C        ;controlla la sintassi
9C17    F0 03       BEQ $9C1C     ;
9C19    4C 08 AF    JMP $AF08     ;'SINTAX ERROR
9C1C    A0 00       LDX 00        ;
9C1E    20 73 00    JSR $0073     ;
9C21    F0 F6       BEQ $9C19     ;
9C23    A5 7A       LDA $7A       ;
9C25    C9 05       CMP 05        ;
9C27    D0 F0       BNE $9C19     ;
9C29    A9 00       LDA 00        ;
9C2B    85 7A       STA $7A       ;
9C2D    20 7C A5    JSR A57C      ;'TOKENISATION ROUTINE
9C30    A5 2B       LDA $2B       ;
9C32    85 F9       STA $F9       ;ind.prima linea in $F9,FA
9C34    A5 2C       LDA $2C       ;
9C36    85 FA       STA $FA       ;
9C38    A9 FF       LDA FF        ;
9C3A    8D 4F 9A    STA $9A4F     ;
9C3D    A2 00       LDX 00        ;
9C3F    A0 01       LDY 01        ;
9C41    B1 F9       LDA ($F9),Y   ;se HB link pross.linea =0
9C43    D0 06       BNE $9C4B     ;e' finito il prog. Basic
9C45    A9 00       LDA 00        ;
9C47    8D 4F 9A    STA $9A4F     ;
9C4A    60          RTS           ;
9C4B    C8          INY           ;
9C4C    A5 F9       LDA $F9       ;---------------------------
9C4E    85 5F       STA $5F       ;
9C50    A5 FA       LDA $FA       ;$5F,60 = inizio linea
9C52    85 60       STA $60       ;
9C54    B1 F9       LDA ($F9),Y   ;---------------------------
9C56    85 14       STA $14       ;
9C58    C8          INY           ;$14,15 = numero linea
9C59    B1 F9       LDA ($F9),Y   ;
9C5B    85 15       STA $15       ;
9C5D    C8          INY           ;---------------------------
9C5E                              ;controlla se nella linea in
9C5E                              ;esame c'è il testo richiesto
9C5E    BD 05 02    LDA $0205,X   ;'caratt. dal buffer
9C61    F0 17       BEQ $9C7A     ;'OK trovato!
9C63    C9 22       CMP 22        ;'confronta con "
9C65    D0 03       BNE $9C6A     ;'non uguali
9C67    E8          INX           ;'salta le "
9C68    D0 F4       BNE $9C5E     ;
9C6A    B1 F9       LDA ($F9),Y   ;'carattere dalla linea
9C6C    F0 11       BEQ $9C7F     ;'fine linea
9C6E    DD 05 02    CMP $0205,X   ;'confr. con buffer
9C71    D0 03       BNE $9C76     ;'caratteri diversi
9C73    C8          INX           ;'car.uguali:pross.car.buff.
9C74    D0 E7       BNE $9C5D     ;'prossimo car. linea
9C76    A2 00       LDX 00        ;'primo car. buffer
9C78    F0 E3       BEQ $9C5D     ;'prossimo car. linea
9C7A    A9 00       LDA 00        ;
9C7C    20 BD A6    JSR $A6BD     ;'lista la linea
9C7F    A0 00       LDY 00        ;---------------------------
9C81    B1 F9       LDA (F9),Y *  ;carica in $F9,FA
9C83    AA          TAX           ;l'indirizzo della
9C84    C8          INY           ;prossima linea da
9C85    B1 F9       LDA ($F9),Y   ;da analizzare
9C87    85 FA       STA $FA       ;
9C89    8A          TXA           ;
9C8A    85 F9       STA $F9       ;
9C8C    18          CLC           ;
9C8D    90 AE       BCC $9C3D     ;'nuova linea
9C8F                              ;NUOVA ERROR ROUTINE
9C8F                              ;---------------------------
9C8F    8A          TXA           ;'se X>=80 solo stampa READY
9C90    10 09       BPL $9C9B     ;'se X<80 stampa errore
9C92    AD 4F 9A    LDA $9A4F     ;
9C95    F0 01       BEQ $9C98     ;
9C97    60          RTS           ;
9C98    4C 74 A4    JMP $A474     ;'stampa READY
9C9B    AA          TAX           ;
9C9C    A5 7A       LDA $7A       ;---------------------------
9C9E    8D 42 9A    STA $9A42     ;
9CA1    A5 7B       LDA $7B       ;memorizza indirizzo errore
9CA3    8D 43 9A    STA $9A43     ;
9CA6    A5 39       LDA $39       ;---------------------------
9CA8    8D 40 9A    STA $9A40     ;
9CAB    A5 3A       LDA $3A       ;memorizza numero linea err.
9CAD    8D 41 9A    STA $9A41     ;
9CB0    8A          TXA           ;
9CB1    4C 3A A4    JMP $A43A     ;ERROR ROUTINE NORMALE
9CB4                              ;NUOVA PRINT TOKENS ROUTINE
9CB4    AA          TAX           ;---------------------------
9CB5    CE 4E 9A    DEC $9A4E     ;'contatore punto errore
9CB8    F0 04       BEQ $9CBE     ;'raggiunto punto errore
```

```
9CBA  84        TXA          ;
9CBB  4C 1A A7  JMP $A71A    ;'torna alla routine origin.
9CBE  A9 3F     LDA 3F       ;
9CC0  20 D2 FF  JSR $FFD2    ;stampa "?"
9CC3  18        CLC          ;
9CC4  70 F4     BCC $9CBA    ;-----------------------------
9CC6                         ;
9CC6                         ;ROUTINE HELP
9CC6                         ;sntx : HELP
9CC6  20 73 00  JSR $0073    ;-----------------------------
9CC9  F0 03     BEQ $9CCE    ;controllo sintassi
9CCB  4C 08 AF  JMP $AF08    ;'SINTAX ERROR
9CCE  AD 40 9A  LDA $9A40    ;-----------------------------
9CD1  85 14     STA $14      ;
9CD3  AD 41 9A  LDA $9A41    ;$14,15 = num.linea errore
9CD6  C9 FF     CMP FF       ;
9CD8  D0 01     BNE $9CDB    ;
9CDA  60        RTS          ;
9CDB  85 15     STA $15      ;-----------------------------
9CDD  A9 FF     LDA FF       ;abilita il RTS nella
9CDF  8D 4F 9A  STA $9A4F    ;routine di LIST
9CE2  A9 B4     LDA B4       ;-----------------------------
9CE4  8D 06 03  STA $0306    ;modifica il vettore della
9CE7  A9 9C     LDA 9C       ;Print Tokens Routine
9CE9  8D 07 03  STA $0307    ;-----------------------------
9CEC  20 13 A6  JSR $A613    ;'cerca la linea nel progr.
9CEF  38        SEC          ;-----------------------------
9CF0  AD 42 9A  LDA$9A42     ;
9CF3  E5 5F     SBC 5F       ;calcola il punto in cui si
9CF5  38        SEC          ;e' verificato l'errore
9CF6  E9 04     SBC 04       ;
9CF8  8D 4E 9A  STA $9A4E    ;-----------------------------
9CFB  20 BD A6  JSR $A6BD    ;lista la linea in esame
9CFE  A9 1A     LDA 1A       ;-----------------------------
9D00  8D 06 03  STA $0306    ;
9D03  A9 A7     LDA A7       ;rimette il valore originale
9D05  8D 07 03  STA $0307    ;nel vettore Print Tokens.
9D08  A9 00     LDA 00       ;-----------------------------
9D0A  8D 4F 9A  STA $9A4F    ;disabilita il RTS nella
9D0D  60        RTS          ;routine di LIST
9D0E                         ;ROUTINE INPUT DUE PARAMETRI
9D0E                         ;-----------------------------
9D0E  A0 00     LDY 00       ;i parametri vengono
9D10  8C 45 9A  STY $9A45    ;memorizzati in $9A44,9A45
9D13  8C 47 9A  STY $9A47    ; e in $9A46,9A47
9D16  A2 0A     LDX 0A       ;
9D18  20 73 00  JSR $0073    ;
9D1B  D0 07     BNE $9D24    ;-----------------------------
9D1D  8E 44 9A  STX $9A44    ;i due parametri = 000A
9D20  8E 46 9A  STX $9A46    ;per default
9D23  60        RTS          ;-----------------------------
9D24  90 06     BCC $9D2C    ;'c'è il primo parametro
9D26  84 15     STY $15      ;non c'è il primo parametro
9D28  86 14     STX $14      ;viene assegnato per default
9D2A  D0 03     BNE $9D2F    ;-----------------------------
9D2C  20 6B A9  JSR $A96B    ;prende il numero dal buffer
9D2F                         ;Basic e lo tiene in $14,15
9D2F  C9 2C     CMP 2C       ;-----------------------------
9D31  F0 07     BEQ $9D3A    ;controllo che il separatore
9D33  C9 2D     CMP 2D       ;sia un "," o un "-"
9D35  F0 03     BEQ $9D3A    ;
9D37  4C 08 AF  JMP $AF08    ;'SINTAX ERROR
9D3A  A5 14     LDA $14      ;-----------------------------
9D3C  8D 44 9A  STA $9A44    ;
9D3F  A5 15     LDA $15      ;memorizza primo parametro
9D41  8D 45 9A  STA $9A45    ;
9D44  20 73 00  JSR $0073    ;-----------------------------
9D47  F0 07     BEQ $9D50    ;manca il secondo parametro
9D49  B0 EC     BCS $9D37    ;non e' un numero
9D4B  20 6B A9  JSR $A96B    ;prende il secondo parametro
9D4E  B0 06     BCS $9D56    ;-----------------------------
9D50  85 15     STA $15      ;valore di default per il
9D52  A9 0A     LDA 0A       ;secondo parametro
9D54  85 1414   STA $14      ;-----------------------------
9D56  A5 15     LDA $15      ;
9D58  8D 47 9A  STA $9A47    ;memorizza secondo parametro
9D5B  A5 14     LDA $14      ;
9D5D  8D 46 9A  STA $9A46    ;
9D60  60        RTS          ;
9D61                         ;ROUTINE AUTO
9D61                         ;sntx : AUTO(a,)(b)
9D61  68        PLA          ;-----------------------------
9D62  68        PLA          ;
9D63  20 0E 9D  JSR $9D0E    ;prende i due parametri dalla
9D66  AD 46 9A  LDA $9A46    ;linea di comando e se b=0
9D69  0D 47 9A  ORA $9A47    ;passa a disabilitare l'AUTO
9D6C  F0 0D     BEQ $9D7B    ;-----------------------------
9D6E  A9 88     LDA 88       ;abilita l'AUTO cambiando il
9D70  8D 02 03  STA $0302    ;vettore della :
9D73  A9 9D     LDA $9D      ;HANDLE BASIC LINE ROUTINE
9D75  8D 03 03  STA $0303    ; ora = $9D88
9D78  6C 02 03  JMP ($0302)  ;-----------------------------
9D7B  A9 50     LDA 50       ;disabilita l'AUTO riportando
9D7D  8D 02 03  STA $0302    ;il vettore al valore preced.
9D80  A9 9A     LDA 9A       ;        $9A50
9D82  8D 03 03  STA $0303    ;
9D85  4C 74 A4  JMP $A474    ;stampa READY
9D88                         ;ESTENSIONE DELLA HANDLE
9D88                         ;BASIC LINE ROUTINE
9D88  AC 44 9A  LDY $9A44    ;-----------------------------
9D8B  AD 45 9A  LDA $9A45    ;'carica numero linea
9D8E  20 91 B3  JSR $B391    ;'converte da int. in float.
9D91  20 DD BD  JSR $BDDD    ;'converte float. in ASCII
9D94  A0 01     LDY 01       ;-----------------------------
9D96  B9 00 01  LDA $0100,Y  ;
9D99  F0 06     BEQ $9DA1    ;copia il numero in ASCII nel
9D9B  99 76 02  STA $0276,Y  ;buffer della tastiera
9D9E  C8        INY          ;
9D9F  D0 F5     BNE $9D96    ;
9DA1  A9 20     LDA 20       ;
9DA3  99 76 02  STA $0276,Y  ;
9DA6  84 C6     STY $C6      ;num.car. nel buffer tast.
9DA8  18        CLC          ;-----------------------------
9DA9  AD 46 9A  LDA $9A46    ;
9DAC  6D 44 9A  ADC $9A44    ;calcola il prossimo numero
9DAF  8D 44 9A  STA $9A44    ;di linea
```

PERSONAL SOFTWARE

Programmer's tool kit per Commodore 64

Seguito listato 1.

```
9DB2   AD 47 9A   LDA $9A47      ;
9DB5   6D 45 9A   ADC $9A45      ;
9DB8   8D 45 9A   STA $9A45      ;
9DBB   B0 BE      BCS $9D7B      ;disabilita AUTO
9DBD   4C 50 9A   JMP $9A50      ;prosegue con la normale
9DC0                             ;rout.di HANDLE
9DC0                             ;ROUTINE TRACE
9DC0                             ;sntx : TRON(,STEP/S) TROFF
9DC0   A9 00      LDA 00         ;---------------------------
9DC2   8D 49 9A   STA $9A49      ;
9DC5   20 73 00   JSR $0073      ;'primo carattere dopo TRO
9DC8   C9 46      CMP 46         ;'="F" ?
9DCA   F0 41      BEQ $9E0D      ;'se si va a TROFF
9DCC   C9 4E      CMP 4E         ;'="N" ?
9DCE   F0 03      BEQ $9DD3      ;'se si va a TRON
9DD0   4C 08 AF   JMP $AF08      ;'SINTAX ERROR
9DD3                             ;---------------------------
9DD3   20 73 00   JSR $0073      ;TRON ROUTINE
9DD6   F0 10      BEQ $9DE8      ;'no STEP
9DD8   C9 2C      CMP 2C         ;controlla se segue ",S"
9DDA   D0 F4      BNE $9DD0      ;altrimenti errore
9DDC   20 73 00   JSR $0073      ;
9DDF   C9 53      CMP 53         ;
9DE1   D0 ED      BNE $9DD0      ;
9DE3   A9 01      LDA 01         ;attiva funzione di STEP
9DE5   8D 49 9A   STA $9A49      ;
9DE8   A0 22      LDY 22         ;---------------------------
9DEA   A2 06      LDX 06         ;
9DEC   A9 A0      LDA A0         ;crea una finestra sullo
9DEE   99 00 04   STA 0400,Y     ;schermo
9DF1   AD 86 02   LDA $0286      ;'colore corrente
9DF4   99 00 D8   STA $D800,Y    ;
9DF7   C8         INY            ;
9DF8   CA         DEX            ;
9DF9   D0 F1      BNE $9DEC      ;
9DFB   98         TYA            ;
9DFC   18         CLC            ;
9DFD   69 22      ADC 22         ;
9DFF   A8         TAY            ;
9E00   90 E8      BCC $9DEA      ;---------------------------
9E02   A9 18      LDA 18         ;aggiorna vettore routine
9E04   8D 08 03   STA $0308      ;BASLINK STATEMENT
9E07   A9 9E      LDA 9E         ;  in $9E18
9E09   8D 09 03   STA $0309      ;
9E0C   60         RTS            ;
9E0D   A9 E4      LDA E4         ;---------------------------
9E0F   8D 08 03   STA $0308      ;TROFF disabilita il TRACE
9E12   A9 A7      LDA A7         ;riportando il vettore della
9E14   8D 09 03   STA $0309      ;BASLINK STATEMENT ROUTINE al
9E17   60         RTS            ;valore originale: $A7E4
9E18                             ;NUOVA BASLINK STAT.ROUTINE
9E18                             ;---------------------------
9E18   A5 3A      LDA $3A        ;se si tratta di una linea
9E1A   C9 FF      CMP FF         ;eseguita in modo diretto
9E1C   D0 03      BNE $9E21      ;passa alla routine normale
9E1E   4C E4 A7   JMP $A7E4      ;'BASLINK STAT.ROUTINE
9E21   A5 39      LDA $39        ;---------------------------
9E23   A4 3A      LDY $3A        ;controlla se è cambiato il
9E25   CD 4A 9A   CMP $9A4A      ;numero di linea, se è lo
9E28   D0 05      BNE $9E2F      ;stesso salta alla routine
9E2A   CC 4B 9A   CPY $9A4B      ; BASLINK normale.
9E2D   F0 EF      BEQ $9E1E      ;
9E2F   8D 4A 9A   STA $9A4A      ;---------------------------
9E32   8C 4B 9A   STY $9A4B      ;'memorizza nuovo num. linea
9E35   AD 49 9A   LDA $9A49      ;'funzione STEP attivata ?
9E38   F0 07      BEQ $9E41      ;'NO : salta la routine seg.
9E3A   EA         NOP            ;
9E3B   A5 C5      LDA $C5        ;attende che venga premuto
9E3D   C9 40      CMP 40         ;un tasto
9E3F   F0 F9      BEQ $9E3A      ;
9E41   A9 4A      LDA 4A         ;---------------------------
9E43   85 52      STA $52        ;
9E45   A9 04      LDA 04         ;esegue lo scroll della
9E47   85 51      STA $51        ;finestra verso l'alto
9E49   85 53      STA $53        ;
9E4B   A9 22      LDA 22         ;
9E4D   85 50      STA $50        ;
9E4F   A0 05      LDY 05         ;
9E51   B1 52      LDA ($52),Y    ;
9E53   91 50      STA ($50),Y    ;
9E55   88         DEY            ;
9E56   10 F9      BPL $9E51      ;
9E58   18         CLC            ;
9E59   A5 50      LDA $50        ;
9E5B   69 28      ADC 28         ;
9E5D   85 50      STA $50        ;
9E5F   A5 52      LDA $52        ;
9E61   69 28      ADC 28         ;
9E63   85 52      STA $52        ;
9E65   90 E8      BCC $9E4F      ;---------------------------
9E67   A9 22      LDA 22         ;aggiorna il colore dei
9E69   85 50      STA $50        ;caratteri nella finestra
9E6B   A9 D8      LDA D8         ;al colore corrente ($0286)
9E6D   85 51      STA $51        ;
9E6F   AD 86 02   LDA $0286      ;
9E72   A0 05      LDY 05         ;
9E74   91 50      STA ($50),Y    ;
9E76   88         DEY            ;
9E77   10 FB      BPL $9E74      ;
9E79   18         CLC            ;
9E7A   A5 50      LDA $50        ;
9E7C   69 28      ADC 28         ;
9E7E   85 50      STA $50        ;
9E80   90 ED      BCC $9E6F      ;---------------------------
9E82   A5 3A      LDA $3A        ;converte il numero di linea
9E84   A4 39      LDY $39        ;da intero in ASCII
9E86   20 91 B3   JSR $B391      ;
9E89   20 DD BD   JSR $BDDD      ;
9E8C   A9 A3      LDA A3         ;---------------------------
9E8E   8D EA 04   STA $04EA      ;mette  il numero di linea
9E91   A0 01      LDY 01         ;in reverse direttamente
9E93   B9 00 01   LDA $0100,Y    ;nella mappa video
9E96   F0 08      BEQ $9EA0      ;
9E98   09 80      ORA 80         ;
9E9A   99 EA 04   STA $04EA,Y    ;
9E9D   C8         INY            ;
9E9E   D0 F3      BNE $9E93      ;
9EA0   A9 A0      LDA A0         ;
9EA2   C0 07      CPY 07         ;
9EA4   F0 06      BEQ $9EAC      ;
9EA6   99 EA 04   STA $04EA      ;
9EA9   C8         INY            ;
9EAA   D0 F6      BNE $9EA2      ;---------------------------
9EAC   4C E4 A7   JMP $A7E4      ;torna alla rout.BASLINK
9EAF                             ; ORIGINALE
9EAF                             ;ROUTINE DELETE
9EAF                             ;sntx : DELETE(a,)(b)
9EAF   68         PLA            ;---------------------------
9EF0   68         PLA            ;
9EF1   20 0E 9D   JSR $9D0E      ;prende parametri dalla linea
9EF4   A2 04      LDA 04         ;di comando
9EF6   BD 43 9A   LDA $9A43,X    ;---------------------------
9EF9   95 F6      STA $F6,X      ;copia i parametri in
9EFB   CA         DEX            ;  $F7,F8  e  $F9,FA
9EFC   D0 F8      BNE $9EF6      ;---------------------------
9EFE   A5 FA      LDA $FA        ;
9EC0   D0 0A      BNE $9ECC      ;se 'b' è dato per default
9EC2   A5 F9      LDA $F9        ;allora è posto uguale a FF00
9EC4   C9 0A      CMP 0A         ;
9EC6   D0 04      BNE $9ECC      ;
9EC8   A9 FF      LDA FF         ;
9ECA   85 FA      STA $FA        ;
9ECC   A5 F7      LDA $F7        ;---------------------------
9ECE   85 14      STA $14        ;
9ED0   A5 F8      LDA $F8        ;
9ED2   85 15      STA $15        ;cerca l'indirizzo della pri-
9ED4   20 13 A6   JSR $A613      ;ma linea da cancellare.
9ED7   A9 E1      LDA E1         ;---------------------------
9ED9   8D 02 03   STA $0302      ;cambia il vettore di
9EDC   A9 9E      LDA 9E         ; BASIC WARM START
9EDE   8D 03 03   STA $0303      ;---------------------------
9EE1   A9 00      LDA 00         ;NUOVO BASIC WARM START
9EE3   8D 00 02   STA $0200      ;
9EE6   A0 01      LDY 01         ;---------------------------
9EE8   B1 5F      LDA ($5F),Y    ;controlla se e' finito il
9EEA   F0 19      BEQ $9F05      ;programma BASIC
9EEC   C8         INY            ;---------------------------
9EED   C8         INY            ;
9EEE   B1 5F      LDA ($5F),Y    ;controlla se ha raggiunto
9EF0   C5 FA      CMP $FA        ;l'ultima linea da cancellare
9EF2   F0 04      BEQ $9EF8      ;
9EF4   B0 0F      BCS $9F05      ;
9EF6   90 09      BCC $9F01      ;
9EF8   88         DEY            ;
9EF9   B1 5F      LDA ($5F),Y    ;
9EFB   C5 F9      CMP $F9        ;
9EFD   F0 02      BEQ $9F01      ;
9EFF   B0 04      BCS $9F05      ;
9F01   38         SEC            ;---------------------------
9F02   20 A9 A4   JSR $A4A9      ;cancella la linea puntata
9F05                             ;  $5F,60
9F05   A9 9A      LDA 9A         ;---------------------------
9F06   8D 03 03   STA $0303      ;ristabilisce il vecchio
9F09   A9 50      LDA 50         ; vettore WARM START
9F0B   8D 02 03   STA $0302      ;  $9F50
9F0F   4C 74 A4   JMP $A474      ;'READY
9F12                             ;ROUTINE RENUMBER
9F12                             ;sntx :REN(a,)(b)
9F12                             ;
9F12   20 0E 9D   JSR $9D0E      ;legge  i due parametri
9F15   AD 47 9A   LDA $9A47      ;
9F18   F0 05      BEQ $9F1F      ;se incremento >255 allora:
9F1A   A2 0E      LDX 0E         ;'ILLEGAL QUANTITY ERROR
9F1C   6C 00 03   JMP ($0300)    ;---------------------------
9F1F   AD 46 9A   LDA $9A46      ;se incremento =0 allora IQE
9F22   F0 F6      BEQ $9F1A      ;---------------------------
9F24   A5 2B      LDA $2B        ;
9F26   85 F7      STA $F7        ;legge inizio programma BAS
9F28   A5 2C      LDA $2C        ;
9F2A   85 F8      STA $F8        ;---------------------------
9F2C   A0 01      LDY 01         ;legge inizio prossima
9F2E   B1 F7      LDA ($F7),Y    ;linea Basic
9F30   D0 01      BNE $9F33      ;se = 0 fine programma
9F32   60         RTS            ;
9F33   85 FA      STA $FA        ;
9F35   88         DEY            ;
9F36   B1 F7      LDA (F7),Y     ;
9F38   85 F9      STA $F9        ;
9F3A   C8         INY            ;
9F3B   C8         INY            ;Y-punta il numero di linea
9F3C   AD 44 9A   LDA $9A44      ;legge il nuovo numero e
9F3F   91 F7      STA ($F7),Y    ;assegna alla linea puntata
9F41   C8         INY            ;
9F42   AD 45 9A   LDA $9A45      ;
9F45   91 F7      STA ($F7),Y    ;---------------------------
9F47   A5 F9      LDA $F9        ;
9F49   85 F7      STA $F7        ;punta la prossima linea
9F4B   A5 FA      LDA $FA        ;
9F4D   85 F8      STA $F8        ;
9F4F   18         CLC            ;---------------------------
9F50   AD 44 9A   LDA $9A44      ;calcola il prossimo
9F53   6D 46 9A   ADC $9A46      ;numero di linea
9F56   8D 44 9A   STA $9A44      ;
9F59   AD 45 9A   LDA $9A45      ;
9F5C   69 00      ADC 00         ;
9F5E   8D 45 9A   STA $9A45      ;
9F61   90 C9      BCC $9F2C      ;'prossima linea
9F63   A2 0F      LDX 0F         ;
9F65   6C 00 03   JMP ($0300)    ;OVERFLOW ERROR
```

dal I° gennaio è in edicola

L. 8.000



È in edicola la prima **Guida all'acquisto di libri italiani ed esteri principalmente di informatica, di elettrotecnica ed elettronica, nonché del software applicativo.**

Oltre 350 testi italiani, 1000 stranieri in lingua originale e 900 package applicativi costituiscono l'attuale assortimento.

La guida è il tuo consulente sicuro per orientarsi nel labirinto dell'editoria tecnica, lo strumento ed il servizio essenziale per chi ha compreso l'importanza della tecnologia nel mondo odierno.

Libri di base e didattici per imparare e capire; applicativi per realizzare e coltivare il proprio hobby; pratici per risolvere i problemi dell'attività quotidiana; di elevata specializzazione per migliorare il proprio background professionale o culturale.

Software per Apple, IBM, Texas, Sinclair, TRS, VIC per risolvere i problemi più complessi o, semplicemente, per giocare. Un'ampia gamma di "applicativi" che comprende tra gli altri i più efficienti Data Base, i più completi programmi per l'elaborazione dei testi, i più sofisticati package grafici oltre naturalmente, ai più divertenti programmi ricreativi.



E inoltre, la nuova linea Software TechnoClub, sviluppata in collaborazione con programmatori professionisti, con una gamma di programmi selezionati e convenienti per il tuo home o personal computer.

Acquista la guida in edicola o ordinala direttamente, compilando e spedendo il coupon sottoriportato, unitamente a L. 8.000.

Potrai così prendere visione anche delle modalità per diventare Socio del **TechnoClub** e godere dei numerosi vantaggi che ne derivano tra i quali

La ricezione gratuita di minimo 8 ulteriori numeri di questa guida. Sarai così costantemente aggiornato su tutte le novità editoriali più qualificate e sui package più interessanti ed innovativi per il tuo computer.

Nessun impegno di acquisto durante il periodo di adesione. Scelta libera e senza vincoli di minimi quantitativi di acquisto durante il periodo di adesione, potendo così ordinare ciò che si vuole, quando si vuole.

Convenienza certa. I testi italiani sono scontati del 10% circa rispetto al prezzo di copertina. Particolarmente vantaggiosi risultano i prezzi dei libri esteri e del software.

La tessera TechnoClub. Il documento personale che dà diritto a sconti speciali su diversi articoli acquistati presso negozi convenzionati.

Oltre 5.000 Soci hanno già aderito al TechnoClub. Attendiamo anche te.

Ordino il primo numero della **Guida all'acquisto di libri e software** e

☐ allego L. 8.000 (in contanti o francobolli)

☐ allego assegno di L. 8.000

Nome ______________________

Cognome ______________________

Via ______________________

Città ______________________ C.A.P. ________ Prov. ______

Data ______________________

Firma ______________________

CEDOLA da compilare e spedire in busta chiusa a Gruppo Editoriale Jackson srl Via Rosellini, 12 20124 Milano

File analyzer

Un indispensabile strumento di lavoro per la manipolazione dei dati, scritto in BASIC Applesoft per Apple II

di *Sergio Orlando*

Introduzione

Questo programma è rivolto a tutti coloro, professionisti e hobbisti, i quali per qualche motivo vogliono intervenire sui dati registrati in file di tipo TEXT (T).

Le applicazioni di questo programma sono molteplici. Esso si può rivelare utile in diverse circostanze: può essere utile al professionista che vuole intervenire su un archivio di cui non conosce la struttura: tramite il File Analyzer si può studiare tale struttura. Può essere anche usato per redigere una documentazione sulla struttura del record, per cambiarne il contenuto, per creare facilmente dei file eseguibili con l'istruzione EXEC, come programma didattico per gli hobbisti i quali possono studiare le più comuni strutture di file.

Per ciò che concerne l'assistenza il F.A. si manifesta poi in tutta la sua potenza in quanto permette interventi veloci su tutti gli archivi, facendo risparmiare una enorme mole di tempo e di lavoro.

Preliminari

Mettiamoci nel caso di dover lavorare su un file di cui conosciamo solo il nome.

```
<cr>
\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\VISICALC<cr>
ASM<cr>
PERSONAL_SOFT.<cr>
S<cr>
GENERAL_PURPOSE<cr>
2.2<cr>
____________________<cr>
APPLE_WRITER<cr>
ASM<cr>
APPLE<cr>
S<cr>
WORD_PROCESSING<cr>
2.4<cr>
__________________________<cr>
APPLE_WRITER_IT.<cr>
ASM<cr>
APPLE<cr>
N<cr>
APPLE_WRITER_IN_ITALIANO<cr>
2.4<cr>
__________<cr>
PRO_WRITER<cr>
ASM<cr>
IRET<cr>
S<cr>
DIV.IN
NUMERO CAMPI = 26
```

Figura 1. *Ecco come si presentano i primi 300 byte di un file da studiare. L'identificazione dei campi è immediata e così pure la struttura di un singolo record. Il numero dei campi segnalato alla fine, in questo caso, non ha alcun valore.*

Per poter intervenire su un file prima di tutto bisogna conoscerne la struttura. Ora, poichè il DOS dell'Apple non permette di esaminare un file di tipo TEXT, né di conoscere la lunghezza di un record in un file random, né di sapere se un file è sequenziale o ad accesso casuale, era necessario fare un programma che permettesse di risalire, seppure non automaticamente, a queste informazioni.

Un metodo per sapere se il file in esame sia random o sequenziale è quello di leggerlo dall'inizio, come verrà specificato nel seguito. Esaminandone il contenuto, è facile vedere che tipo di informazioni vi sono memorizzate, se c'è una certa periodicità nel tipo di informazioni (nel qual caso il file è random) e la eventuale lunghezza esatta del record.

Per esempio nella figura 1 si vedono i risultati ottenuti seguendo il metodo indicato: si tratta di un archivio che contiene i programmi facenti parte della mia biblioteca personale, con alcune informazioni sui programmi stessi. Avendo specificato una lunghezza di 300 byte, è facile constatare che il file inizia con un simbolo di fine campo «cr» seguito da 67 "null"; poi comincia una certa periodicità da cui si comprende che il file è random e che il record comincia col nome di un programma e termina con dei blanks seguiti dal simbolo di fine campo (che in questo caso è anche la fine del record). La lunghezza del record è di 68 caratteri.

Un altro metodo è quello di far girare il programma che usa l'archivio in esame in MON C, I, O, il che permette di trarre le informazioni desiderate, cioè il tipo del file, la eventuale lunghezza del record, ecc.

Uso del programma

E veniamo al funzionamento del

Parametro	Fase del programma (vedi figura 2)	Lunghezza max. in byte
lunghezza del record	1.1; 1.2	1000
n° di caratteri da visualizzare	2.1	720
n° di caratteri da modificare	2.2	360
n° di caratteri da registrare su disco	2.3	720

Tabella 1. *Tabella dei valori massimi ammessi dal programma per alcuni parametri.*

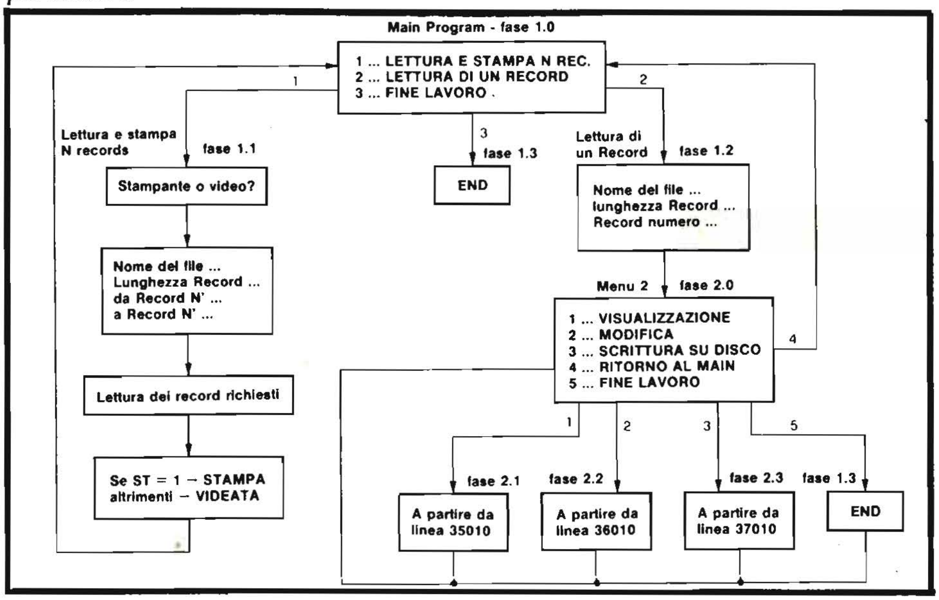


Figura 2. *Si vede la gerarchia dei menu e le possibili scelte che offre il programma nelle sue fasi.*

programma. I limiti di alcuni parametri da esso imposti nelle varie fasi sono riepilogati nella tabella 1; tali limiti sono determinati solo dalla possibilità di visualizzazione e quindi facilmente modificabili.

Per capire la struttura generale del programma basta guardare lo schema a blocchi di figura 2.

In esso è evidenziata la gerarchia dei menu e le possibili scelte che via via si presentano all'utilizzatore.

Lettura e scrittura N record

Partendo dal primo riquadro si vede che le possibili scelte sono 2, oltre naturalmente quella di fine lavoro. La "LETTURA E SCRITTURA N REC." permette di leggere un numero qualunque di record da un file e di visualizzarli uno per volta sul video oppure di stamparli tutti.

È bene spendere due parole per le convenzioni dei simboli.

Il carattere nullo viene visualizzato tramite un back slash (\); il carriage return viene visualizzato come una R in reverse su video e con la scrittura «cr» in stampa; lo spazio bianco viene invece segnalato con un underline (—) per facilitarne il conteggio in caso di sequenza. Poichè, però, le varie stampanti possono rispondere in modo diverso allo stesso codice, all'inizio della stampa è posta una legenda che segnala come vengono stampati i tre codici suddetti (null, carriage return e blank), come è visibile nella figura 3.

Il nome del file va specificato la prima volta che viene richiesto; le volte successive, per confermarlo, basta premere il tasto RETURN. La lunghezza e i record di inizio e di fine vanno sempre specificati.

La fase 1 può essere utile per avere una visione d'insieme del file.

Stampando alcuni record si ottiene una documentazione che specifica il nome del file, la lunghezza del record, il numero di campi e ne mostra il contenuto. Un esempio è visibile nella figura 3.

Lettura di un record

La fase 2 - LETTURA DI UN RECORD - è quella che bisogna richiamare qualora si vogliano fare delle modifiche al contenuto di un record o più in generale di un qualunque file di tipo TEXT.

La massima lunghezza del record ammessa è di 1000 caratteri che dovrebbe essere sufficiente in quasi tutti i casi; se non bastasse con qualche artificio si può comunque accedere a record anche più lunghi.

Poichè questo programma è stato concepito inizialmente per i soli file random, richiede sempre la lunghezza del record o il numero del record che si vuole leggere. Ciò per il fatto che sull'Apple i file sequenziali sono usati molto poco a causa di alcuni bachi nel DOS che ne rendono problematico l'utilizzo. Comunque questo fatto non è in alcun modo un impedimento, in quanto si riescono a leggere e modificare tranquillamente anche i file sequenziali.

Supponiamo ad esempio di voler leggere un file sequenziale di 3000 caratteri; per il modo in cui opera il DOS dell'Apple, specificando una lunghezza record di 500 caratteri si può accedere ai primi 500 byte del file leggendo il record 0, ai secondi 500 leggendo il record 1, ai successivi con la lettura del record 2 e così via.

In questo modo è anche possibile apportare le modifiche volute ai dati in modo del tutto trasparente al DOS che non fa nessuna distinzione tra file sequenziali e random.

Ritornando al programma, supponiamo di aver letto un record: le operazioni che possiamo fare adesso sono di visualizzazione (solo su video), di modifica e di registrazione permanente su disco delle modifiche apportate al record.

Visualizzazione

La visualizzazione non presenta problemi: bisogna specificare il byte di inizio (si suppone che il primo byte del record sia il N° 1 e non il N° 0 come fa il DOS) e il byte di fine; la visualizzazione questa volta è sequenziale, cioè non c'è un "a capo" ad ogni carriage return come succedeva nella fase 1 vista prima. La simbologia, invece, è la stessa.
In alto, sullo schermo, è visibile la numerazione dei byte; se, per esempio, noi vogliamo vedere il record da byte 24 a byte 48, la numerazione in alto partirà da 4 (unità del 24) e terminerà ad 8 e sarà visualizzata come segue:

4567890123456789012345678

In questo modo è abbastanza facile selezionare la parte del record che interessa e fare i conti per le successive eventuali operazioni.

Modifica

La fase di modifica divide lo schermo in due metà: nella parte superiore visualizza la porzione di record da noi selezionata; in quella inferiore mostra le nostre modifiche. È possibile confermare il carattere già esistente premendo RETURN o modificarlo premendo il tasto voluto. Per introdurre un «cr» - codice 13 - bisogna premere CTRL-R.
È possibile modificare più volte il record finchè non si raggiunge il risultato desiderato.

Registrazione su disco

A questo punto si può memorizzare il record modificato su disco, il che rende effettive le modifiche.
Anche nella fase di registrazione su disco, la fase 3, bisogna specificare il byte di inizio e quello di fine.
Dopo quest'ultima risposta, però, il computer partirà come un razzo e farà delle cose strane, modificando

```
<cr> = SIMBOLO DI FINE CAMPO
     = SPAZIO BIANCO
\    = CARATTERE NULLO

FILE MSI.FL          DA RECORD 24   A RECORD 29          LUNGHEZZA RECORD 63

[illegible]
[illegible]

MPATCH<cr>
ASM<cr>
<cr>
N<cr>
METTE_A_NUDO_LA_MEMORIA<cr>
2.1<cr>
---                    <cr>

NUMERO CAMPI = 7


L.I.S.A.<cr>
ASM<cr>
PROGRAMMA<cr>
N<cr>
ASSEMBLER_INTERATTIVO<cr>
2.3<cr>
_________________<cr>

NUMERO CAMPI = 7


INTEGER*<cr>
ASM<cr>
<cr>
N<cr>
SIMULATORE_INTEGER_BASIC<cr>
2.3<cr>
____________________________<cr>

NUMERO CAMPI = 7


FUNCTION_PLOT<cr>
<cr>
<cr>
N<cr>
GRAFICI_DI_FUNZIONI<cr>
4.1<cr>
______________________________<cr>

NUMERO CAMPI = 7


CHARACTER_GENER.<cr>
ASM<cr>
<cr>
N<cr>
CARATT._IN_ALTA_RISOLUZ.<cr>
4.1<cr>
________________<cr>

NUMERO CAMPI = 7
```

Figura 3. *Ecco come si presenta la stampa finale di 5 record di un file. Si nota la legenda in alto, il nome del file, il numero dei record stampati, la lunghezza del record (impostata dall'utente), due righe di cifre che danno la posizione di colonna - 1 ed il contenuto dei 5 record.*
Il numero dei campi del record (7), questa volta è esatto. Si vede che la stampa può servire da documentazione.

probabilmente il byte di fine impostato dall'utente.

Perchè tutto questo?

Per capirlo basta ricordare che quando si scrive una parola su disco, il DOS mette un segno di riconoscimento per indicare la fine della parola stessa: tale segno è un «cr» - codice 13 -.

Ora supponiamo di essere nella seguente situazione:

...MARCO«cr»...

e di voler cambiare MARCO con MARIO. È sufficiente cambiare la "C" con la "I". Se l'operatore memorizzasse su disco solo la "I", il DOS metterebbe dopo tale carattere un «cr», per cui sul disco ci sarebbe questa situazione:

...MARI«cr» «cr»...

che non è quella voluta.

Il programma allora provvede a cercare dopo la "I" il prossimo «cr» che nel caso in esame si trova dopo la "O" e scrive su disco, dopo avere chiesto l'ultimo OK all'utente, tutta la stringa di caratteri che compaiono sul video, tranne l'ultimo «cr». Nell'esempio di prima il programma visualizzerebbe:

IO «cr»

e scriverebbe su disco la stringa "IO", mentre il «cr» verrebbe aggiunto automaticamente dal DOS ottenendo in totale la giusta stringa:

...MARIO «cr»...

Da notare che non si possono memorizzare stringhe contenenti caratteri nulli, il che non è in nessun caso una limitazione.

Descrizione del programma

Nella figura 4 è riportata una mappa del programma con i numeri di linea da cui cominciano le varie sezioni evidenziate. La prima istruzione porta il controllo alla linea 60010 dove ci sono le inizializzazioni di alcune variabili il cui valore non viene più modificato durante l'esecuzione del programma e si memorizza una routine utile quando l'ONERR GOTO è attivo.

500	Vai a inizializzazioni (60010)
1010	Routine di uso comune
30010	Lettura e visualizzazione n. Record
32010	Lettura di un record
35010	Visualizzazione
36010	Modifica
37010	Registrazione su disco
38010	Ritorno al main
60010	Inizializzazione variabili
60510	Main programma
62010	END programma

Figura 4. *Mappa del programma.*

Il main program propone all'utente la scelta fra le tre alternative di:

1. Visualizzare o stampare alcuni record.
2. Leggere un solo record per poi apportarvi alcune modifiche.
3. Terminare il programma.

La prima scelta porta il controllo alla linea 30010, la seconda alla linea 32010, la terza alla linea 62010.

Prima di esaminare le tre fasi suesposte è meglio vedere in dettaglio le routine di uso comune che partono dalla linea 1010. Esse verranno usate in varie parti del programma e la loro descrizione iniziale faciliterà le cose in seguito. Si metterà inizialmente il numero di linea da cui comincia la routine, considerando la prima istruzione eseguibile e non il REM che in genere precede la routine stessa. Subito dopo viene specificata la sua funzione ed eventualmente vengono evidenziate le variabili di passaggio.

1010 Disegna la cornice bianca passante per i quattro angoli dello schermo.

1150 Stampa o visualizza sullo schermo il simbolo di fine campo: «cr».

1210 Chiede all'utente se vuole la stampa o se preferisce vedere i risultati sul video; se si vuole la stampa la variabile ST va a 1.

1510 Questa è la routine più importante del programma. Chiede il nome del file, la lunghezza del record e quanti e quali record si vogliono leggere.

Dopo avere controllato la validità dei parametri introdotti dall'utente, alla linea 1555 testa la variabile booleana di stampa ST: se è ad 1 va alla subroutine 8010 dove si provvede a stampare l'intestazione visibile nella figura 3.

Alla linea 1560 comincia il ciclo di lettura dei record specificati; per quanto riguarda il significato delle variabili ci si può riferire alla tavola delle variabili di figura 5.

Alla linea 1665 il test su SL verifica se questa routine è stata chiamata dalla fase 1 - LETTURA E STAMPA N REC. - (SL = 0), oppure dalla fase 2 - LETTURA DI UN RECORD - (SL = 1), nel qual caso la routine termina.

In riga 1670 si chiama, se ST = 1, la routine che stampa il contenuto del record come si vede in figura 3. Se ST = 0 si esegue invece l'istruzione 1680 che richiama la routine di visualizzazione su video dello stesso record.

7010 Visualizzazione su video del record appena letto (vedi sub. 1510).

8010 Stampa dell'intestazione (vedi sub. 1510).

8210 Stampa del contenuto del record corrente (vedi sub. 1510).

9010 Questa è una routine molto delicata che verifica se ciò che l'utente vuole scrivere su disco, dopo avere apportato delle correzioni al contenuto del record, è lecito oppure produrrebbe degli errori dovuti al peculiare funzionamento del DOS. In particolare la stringa di caratteri che si vuole memorizzare su disco deve terminare con un «cr» e non deve contenere caratteri nulli - codi-

ce 0 -. Nel caso vi siano dei caratteri nulli la variabile ER va ad 1.

20010 Questa è la routine chiamata nel caso che si determini un errore durante la fase di lettura del record dovuto al tentativo di leggere caratteri nulli da disco. Tali caratteri vengono memorizzati nel vettore A$(★) come back slash - codice 92 -.

30010 La fase 1.2 inizializza SL ad 1 (vedi sub. 1510) e poi richiede all'utente di specificare il nome del file, la lunghezza del record ed il numero del record che si vuole leggere. Il formato degli input ed i controlli sui valori digitali dall'operatore sono gli stessi visti all'inizio della subroutine 1510, solo che qui manca l'ultima domanda in quanto si legge un solo record.

Per questo motivo N2 viene posta uguale a N1 (vedi la lista delle variabili per il loro significato) e la subroutine di lettura del record viene chiamata a partire dalla linea 1560 anzichè dalla 1510.

A questo punto, dopo aver letto il record, si presentano all'operatore le possibili scelte tramite il menu 2 che parte da linea 33010. A seconda della risposta data, la linea 33110 passa il controllo ad una delle seguenti fasi:

1. VISUALIZZAZIONE (linea 35010)
2. MODIFICA (36010)
3. REGISTRAZIONE SU DISCO (37010)
4. RITORNO AL MAIN (38010)
5. FINE LAVORO (62010)

35010 La fase di visualizzazione chiede il byte di inizio (B1) e quello di fine (B2), dopo di che fa alcuni controlli per verificare la congruenza dei dati. È da notare che questa fase viene utilizzata come routine anche dalle due fasi successive di modifica (SEL$ = 2) e di registrazione su disco (SEL$ = 3), pertanto le linee 35026, 35038, 35040, 35165 vengono eseguite solo nel caso si provenga da una di esse.

Le linee da 35050 a 35090 provvedono a visualizzare sullo schermo, in alto, una riga di cifre per facilitare

```
A$(*)   V,S,O - RECORD APPENA LETTO DAL FILE
1620 1630 1630 1640 7090 7100 8220 8230 8250 9012 9014 9130 9375
20030 35130 35140 36090 36095 36100 36105 36105 36110 36110 37080
37100 60030

AB$ - S,S,I   CARATTERE DIGITATO DA TASTIERA NELLA FASE DI MODIFICA RECORD
36080 36090 36095 36100 36105 36110

AP$ - S,S,L - CARATTERE DA SCRIVERE NEL FILE
37080 37100 37110

B1 - S,R,I - PUNTA AL BYTE DA CUI INIZIARE LA SCANSIONE DI A$(*)
9012 9120 9360 9370 35020 35025 35026 35035 35036 35038 35060
35060 35120 36075 37060

B2 - S,R,I - PUNTA AL BYTE IN CUI TERMINARE LA SCANSIONE DI A$(*)
9012 9012 9012 9012 9014 9020 9030 9030 9110 9110 9120 9130 9320
9360 9370 9510 35030 35035 35035 35036 35038 35120 36078 37060

BL$   S,S,L - 40 BLANKS
1030 1040 60040

D$ - S,S,L - CHR$(4)
1580 1570 1655 1660 7065 8010 8170 8210 8280 20030 20040 20050
35100 35160 37050 37070 37150 60010

ER - S,B,L - INDICATORE DI ERRORE
9010 9120 9430 35040 37010

HT - S,R,L - TABULATORE ORIZZIONTALE
9010 9210 9320 9340 9510 36050 36070 36075 36090 36090 36090
36095 36095 36095 36100 36100 36100 36105 36105 36105 36110 36110
36110 36200 36200

I - S,R,L - CONTATORE DI CICLO (FOR ... NEXT)
1060 1070 1070 1610 1620 1630 1630 1640 1650 7020 7060 7080 7090
7100 7110 8060 8080 8100 8110 8150 8215 8220 8230 8240 20030
20050 35050 35090 35120 35130 35140 35150 37060 37070 37080 37100
37120

K - S,R,L - CONTATORE DI CICLO (FOR ... NEXT)
7030 7040 7050 8070 8090 8120 8130 8140 9220 9220 9360 9360 9370
9375 9380 9410 9420 9420 35060 35070 35080

LR - S,R,I - LUNGHEZZA RECORD
1520 1525 1580 1610 7080 8030 8215 8250 9020 20040 32050 32055
35026 35035 37050

N1 - S,R,I - PUNTA AL PRIMO RECORD DA LEGGERE
1530 1560 8030 32060 32070

N2 - S,R,I - PUNTA ALL'ULTIMO RECORD DA LEGGERE
1540 1560 8030 32070

NC - S,R,O - NUMERO DEI CAMPI DEL RECORD
1600 1640 1640 8260

NF$ - S,S,I - MEMORIZZAZIONE PERMANENTE DEL NOME DEL FILE (VEDI TF$)
1512 1514 1516 1580 1590 8030 20030 20040 20050 32035 32040 32045
37050 37070

NR - S,R,L - PUNTATORE AL RECORD CORRENTE
1560 1590 1690 20050 37070

PU - S,R,L - PUNTATORE AL CARATTERE CORRENTE DI A$(*)
36075 36078 36090 36095 36100 36105 36105 36110 36110

SEL$ - S,S,I - VARIABILE DI SELEZIONE DA TASTIERA
7150 33100 33110 35026 35038 35040 35165 35165 35170 37020 37030
37040 60550 60560

SL - S,B,L - E' UGUALE A 1 SE SI LEGGE UN SOLO RECORD
1665 32010 60515

ST - S,B,L - E' UGUALE A 1 SE SI VUOLE LA STAMPA
1210 1230 1555 1670 60515

SV$ - S,S,I - VARIABILE TEMPORANEA: SE SV$ = "S" -> ST = 1
1220 1230

TF$ - S,S,I - VARIABILE TEMPORANEA: CONTIENE IL NOME DEL FILE
1510 1512 1514 1516 32030 32035 32040 32045

VT - S,R,L - TABULATORE VERTICALE
9010 9210 9320 9340 9350 9400 9510 36050 36060 36075 36090 36095
36100 36105 36110 36200 36200

END OF VAR. LIST
```

Figura 5. *Lista delle variabili. Per ogni variabile sono riportate tre lettere che indicano: per la prima, m indica matrice, v indica vettore, s indica scalare; per la seconda, r indica reale, i indica intero, b indica booleano, s indica stringa; per la terza, i indica ingresso, o indica uscita, l indica lavoro.*
Segue la descrizione del significato della variabile e tutti i numeri di linea del programma in cui tale variabile è usata.

File analyzer

l'identificazione della posizione esatta di un carattere nel record.

Le linee da 35120 a 35150 provvedono alla visualizzazione vera e propria del record contenuto in A$(★), dopo di che si ritorna al menu 2.

36010 Dopo aver visualizzato il contenuto del record (GOSUB 35010), si passa alla vera e propria fase di modifica. La linea 36080 legge il carattere digitato da tastiera e lo memorizza nella variabile temporanea AB$, dopo di che cominciano una serie di controlli per verificare se il carattere battuto era un carattere speciale (RETURN, CTRL-R o BARRA).

In linea 36110 c'è la memorizzazione definitiva del carattere digitato nel vettore A$(★).

37010 Nella fase di registrazione su disco si provvede alla visualizzazione di ciò che andrà effettivamente scritto sul disco (GOSUB 35010) e si chiede l'ultima conferma all'utente (37020). In caso di risposta affermativa dalla linea 37050 parte la routine di scrittura che avviene byte a byte tramite la variabile AP$. La linea 37080 ritrasforma il carattere underline - codice 95 - in blank, ripristinando la situazione originale.

Conclusioni

Il presente programma è uno strumento molto potente, rivolto soprattutto ai professionisti che devono vedersela frequentemente con file dati, di cui non conoscono la struttura, per apportarvi delle modifiche o per fare ordinaria manutenzione; è facile da usare, perchè sempre guidato da chiare istruzioni e presuppone solo limitate conoscenze di base sui file.

Esso consente di studiare e manipolare i file nel modo voluto e di effettuare tutte le operazioni necessarie (lettura dati, modifica, registrazione su disco) sia su file di tipo sequenziale che su file random. ■

Listato 1. *Il programma BASIC.*

```
10  REM  !INTEGER I,NR,K
500  GOTO 60010
1000  REM QUADRO
1010  INVERSE
1020  HOME
1030  PRINT BL$
1040  VTAB 22: PRINT BL$
1050  VTAB 2
1060  FOR I = 2 TO 21: PRINT " ":
      NEXT
1070  FOR I = 2 TO 21: HTAB 40: VTAB
     I: PRINT " ": NEXT
1080  NORMAL
1090  POKE 33,34
1100  POKE 32,5: VTAB 8
1110  RETURN
1150  PRINT  CHR$ (60) +  CHR$ (9
     9) +  CHR$ (114) +  CHR$ (62
     ): RETURN : REM <CR>
1200  REM STAMPA O VIDEO
1210 ST = 0
1220  PRINT "STAMPANTE O VIDEO (S
     /V) ";: GET SV$: PRINT
1230  IF SV$ = "S" THEN ST = 1
1240  TEXT
1250  RETURN
1500  REM LETTURA RECORD
1510  INPUT "NOME DEL FILE
       ";TF$
1512  IF TF$ = "" AND NF$ = "" THEN
      VTAB 8: GOTO 1510
1514  IF TF$ = "" THEN  VTAB 8: HTAB
     23: PRINT NF$: GOTO 1520
1516 NF$ = TF$
1520  INPUT "LUNGHEZZA RECORD
       ";LR
1525  IF LR > 1000 THEN  VTAB 9: GOTO
     1520
1530  INPUT "DA RECORD N'
       ";N1
1540  INPUT "A RECORD N'
       ";N2
1550  TEXT
1555  IF ST THEN  GOSUB 8010
1560  FOR NR = N1 TO N2
1570  ONERR  GOTO 20000
1580  PRINT D$;"OPEN";NF$;",L";LR

1590  PRINT D$;"READ";NF$;",R";NR

1600 NC = 0
1610  FOR I = 1 TO LR
1620  GET A$(I)
1630  IF A$(I) =  CHR$ (32) THEN
     A$(I) =  CHR$ (95): REM BLAN
     K
1640  IF A$(I) =  CHR$ (13) THEN
     NC = NC + 1
1650  NEXT I
1655  PRINT D$
1660  PRINT D$;"CLOSE"
1662  POKE 216,0
1665  IF SL THEN  RETURN
1670  IF ST THEN  GOSUB 8210: GOTO
     1690
1680  GOSUB 7010
1690  NEXT NR
1700  RETURN
7000  REM VIDEO
7010  HOME
7020  FOR I = 1 TO 4
7030  FOR K = 0 TO 9
7040  PRINT K;
7050  NEXT K
```

Seguito listato 1.

```
7060  NEXT I
7065  PRINT D$
7070  POKE 34,2: VTAB 5
7075  POKE 35,20
7080  FOR I = 1 TO LR
7090  IF A$(I) =  CHR$ (13) THEN
      INVERSE : PRINT  CHR$ (82):
      NORMAL : GOTO 7110
7100  PRINT A$(I);
7110  NEXT I
7120  PRINT
7150  VTAB 22: INPUT "<CR> PER CO
     NTINUARE ";SEL$
7190  TEXT
7200  RETURN
8000  REM STAMPA INTESTAZIONE
8010  PRINT D$;"PR#1"
8020  PRINT  CHR$ (9);"80N"
8022  PRINT  CHR$ (60) +  CHR$ (9
     9) +  CHR$ (114) +  CHR$ (62
     ) + " = SIMBOLO DI FINE CAMP
     O"
8024  PRINT  CHR$ (95) + "     = S
     PAZIO BIANCO"
8026  PRINT  CHR$ (92) + "     = C
     ARATTERE NULLO"
8028  PRINT : PRINT
8030  PRINT "FILE ";NF$; SPC( 10)
     ;"DA RECORD ";N1; SPC( 3);"A
      RECORD ";N2; SPC( 10);"LUNG
     HEZZA RECORD ";LR
8040  PRINT
8050  PRINT ""; SPC( 10);
8060  FOR I = 1 TO 7
8070  FOR K = 1 TO 10
8080  PRINT I;
8090  NEXT K
8100  NEXT I
8110  FOR I = 1 TO 8
8120  FOR K = 0 TO 9
8130  PRINT K;
8140  NEXT K
8150  NEXT I
8160  PRINT
8170  PRINT D$;"PR#0"
8180  RETURN
8200  REM STAMPA RECORD
8210  PRINT D$;"PR#1": PRINT  CHR$
     (9);"80N"
8215  FOR I = 1 TO LR
8220  IF A$(I) =  CHR$ (13) THEN
      GOSUB 1150: GOTO 8240
8230  PRINT A$(I);
8240  NEXT I
8250  IF A$(LR) =  CHR$ (36) THEN
      PRINT
8260  PRINT : PRINT "NUMERO CAMPI
      = ";NC
8270  PRINT : PRINT
8280  PRINT D$;"PR#0"
8290  RETURN
9000  REM VERIFICA
```

Seguito listato 1.

```
9005  REM UP
9010 ER = 0:VT = 9:HT = 10
9012  IF B2 = B1 AND A$(B2) =  CHR$
     (13) THEN B2 = B2 + 1
9014  IF A$(B2) =  CHR$ (13) THEN
      GOSUB 9310: RETURN
9020  IF B2 = LR THEN 9110
9030 B2 = B2 + 1: GOSUB 9510: GOTO
     9014
9100  REM DOWN
9110 B2 = B2 - 1: GOSUB 9510
9120  IF B2 = B1 THEN ER = 1: GOSUB
     9210: RETURN
9130  IF A$(B2) =  CHR$ (13) THEN
      GOSUB 9310: RETURN
9140  GOTO 9110
9200  REM NON SI PUO'
9210  PRINT  CHR$ (7): VTAB VT +
     3: HTAB HT + 6: PRINT "NON S
     I PUO'"
9220  FOR K = 1 TO 1000: NEXT K
9230  RETURN
9300  REM TROVATO
9310  PRINT  CHR$ (7)
9320  VTAB VT: HTAB HT: FLASH : PRINT
     B2;: NORMAL : PRINT " "
9340  VTAB VT + 3: HTAB HT + 6: PRINT
     "ALL RIGHT"
9350  VTAB VT + 6: HTAB 10: INVERSE
     : PRINT "VERIFICA": NORMAL
9360  FOR K = 1 TO (1000 - (B2 -
     B1)): NEXT K
9370  FOR K = B1 TO B2
9375  IF A$(K) =  CHR$ (92) THEN
     9400
9380  NEXT K
9390  RETURN
9400  PRINT  CHR$ (7); CHR$ (7): VTAB
     VT + 6: INVERSE : PRINT "ATT
     ENZIONE";: NORMAL : PRINT "
      "
9405  PRINT : PRINT "ELIMINARE CA
     RATTERI NULLI"
9410  PRINT : PRINT "PRIMO 'NULL'
      IN POSIZIONE ";K
9420  FOR K = 1 TO 3000: NEXT K
9430 ER = 1: RETURN
9500  REM CONTATORE
9510  VTAB VT: HTAB HT: PRINT B2;
     : PRINT " ": RETURN
20000  REM TRATTAMENTO ERRORE
20010  CALL 768
20020 A$(I) =  CHR$ (92): REM  NU
     LL
20030  PRINT D$;"CLOSE";NF$
20040  PRINT D$;"OPEN";NF$;",L";L
     R
20050  PRINT D$;"READ";NF$;",R";N
     R;",B";I
20060  GOTO 1650
30000  REM LETTURA E VISUALIZZAZI
     ONE N RECORDS
```

Seguito listato 1.

```
30010  GOSUB 1010
30020  GOSUB 1210
30030  GOSUB 1010
30040  GOSUB 1510
30050  RETURN
32000  REM LETTURA UN RECORD
32010 SL = 1
32020  GOSUB 1010
32030  INPUT "NOME DEL FILE
        ";TF$
32035  IF TF$ = "" AND NF$ = "" THEN
     VTAB 8: GOTO 32030
32040  IF TF$ = "" THEN  VTAB 8: HTAB
     23: PRINT NF$: GOTO 32050
32045 NF$ = TF$
32050  INPUT "LUNGHEZZA RECORD
        ";LR
32055  IF LR > 1000 THEN  VTAB 9:
     GOTO 32050
32060  INPUT "RECORD NUMERO:
        ";N1
32070 N2 = N1
32080  TEXT
32090  GOSUB 1560
33000  REM MENU 2
33010  GOSUB 1010
33020  PRINT "1...VISUALIZZAZIONE
     RECORD"
33030  PRINT "2...MODIFICA RECORD
     "
33040  PRINT "3...SCRITTURA RECOR
     D SU FILE"
33050  PRINT "4...RITORNO AL MAIN
     "
33060  PRINT "5...FINE LAVORO"
33100  VTAB 18: INPUT "SELEZIONA
     PER NUMERO E <CR> ";SEL$
33105  TEXT
33110  ON  VAL (SEL$) GOTO 35010,
     36010,37010,38010,62010
33130  GOTO 33010
35000  REM VISUALIZZAZIONE
35010  GOSUB 1010
35020  INPUT "DA BYTE: ";B1
35025  IF B1 < 1 THEN  VTAB 8: GOTO
     35020
35026  IF SEL$ = "3" AND B1 = LR THEN
     VTAB 8: GOTO 35020
35030  INPUT "A  BYTE: ";B2
35035  IF B2 > LR OR B2 < B1 THEN
     VTAB 9: GOTO 35030
35036  IF (B2 - B1) >  = 720 THEN
     TEXT : GOTO 35010
35038  IF SEL$ = "2" AND (B2 - B1
     ) >  = 360 THEN  TEXT : GOTO
     35010
35040  IF SEL$ = "3" THEN  GOSUB
     9010: IF ER THEN  TEXT : HOME
     : RETURN
35045  TEXT : HOME
35050  FOR I = 1 TO 4
35060  FOR K = B1 TO B1 + 9
35070  PRINT  RIGHT$ ( STR$ (K),1
     );
```

Seguito listato 1.

```
35080  NEXT K
35090  NEXT I
35100  PRINT D$
35110  POKE 34,2: VTAB 3
35120  FOR I = B1 TO B2
35130  IF A$(I) =  CHR$ (13) THEN
     INVERSE : PRINT  CHR$ (82);
     : NORMAL : GOTO 35150
35140  PRINT A$(I);
35150  NEXT I
35160  PRINT D$
35165  IF SEL$ = "2" OR SEL$ = "3
     " THEN  RETURN
35170  VTAB 22: INPUT "<CR> PER C
     ONTINUARE ";SEL$
35200  TEXT
35210  GOTO 33010
36000  REM MODIFICA RECORD
36010  GOSUB 35010
36020  VTAB 12
36030  HTAB 10: INVERSE : PRINT "
     SEZIONE MODIFICA RECORD": NORMAL

36040  PRINT " <CR> PER CONFERMAR
     E - CTRL/R -> RETURN"
36050  POKE 34,13:VT = 0:HT = 1
36060  VTAB 15 + VT
36070  HTAB HT
36075 PU = B1 + HT - 1 + 40 * VT
36078  IF PU > B2 THEN  TEXT : HOME
     : GOTO 33010
36080  GET AB$
36090  IF AB$ =  CHR$ (13) AND A$
     (PU) =  CHR$ (13) THEN  VTAB
     15 + VT: HTAB HT: INVERSE : PRINT
     "R": NORMAL :HT = HT + 1: GOTO
     36200
36095  IF AB$ =  CHR$ (13) THEN  VTAB
     15 + VT: HTAB HT: PRINT A$(P
     U):HT = HT + 1: GOTO 36200
36100  IF AB$ =  CHR$ (18) THEN A
     $(PU) =  CHR$ (13): VTAB 15 +
     VT: HTAB HT: INVERSE : PRINT
     "R": NORMAL :HT = HT + 1: GOTO
     36200
36105  IF AB$ =  CHR$ (32) THEN A
     $(PU) =  CHR$ (95): VTAB 15 +
     VT: HTAB HT: PRINT A$(PU):HT
      = HT + 1: GOTO 36200
36110 A$(PU) = AB$: VTAB 15 + VT:
      HTAB HT: PRINT A$(PU):HT =
     HT + 1
36200  IF HT > 40 THEN HT = 1:VT =
     VT + 1
36210  GOTO 36060
37000  REM SCRITTURA SU DISCO
37010  GOSUB 35010: IF ER THEN 33
     010
37020  VTAB 22: INPUT "TUTTO OK (
     S/N) ";SEL$
37030  IF  LEFT$ (SEL$,1) = "N" THEN
      TEXT : HOME : GOTO 33010
37040  IF  LEFT$ (SEL$,1) <  > "S
     " THEN 37020
```

File analyzer

Seguito listato 1.

```
37050  PRINT D$;"OPEN";NF$;",L";L
     R
37060  FOR I = B1 TO B2 - 1
37070  PRINT D$;"WRITE";NF$;",R";
     NR;",B";I - 1
37080  IF A$(I) = CHR$ (95) THEN
     AP$ = CHR$ (32): GOTO 37110

37100 AP$ = A$(I)
37110  PRINT AP$
37120  NEXT I
37150  PRINT D$;"CLOSE"
37200  TEXT : HOME : GOTO 33010
38000  REM RITORNO AL MAIN
38010  TEXT
38020  RETURN
60000  REM INIZIALIZZAZIONI
60010 D$ =  CHR$ (4)
60020  POKE 768,104: POKE 769,168
     : POKE 770,104: POKE 771,166
     : POKE 772,223: POKE 773,154
     : POKE 774,72: POKE 775,152:
      POKE 776,72: POKE 777,96
60030  DIM A$(1000)
```

Seguito listato 1.

```
60040 BL$ = "
                           "
60500  REM MAIN
60510  GOSUB 1010
60515 ST = 0:SL = 0
60520  PRINT "1...LETTURA E STAMP
     A N REC."
60530  PRINT "2...LETTURA DI UN R
     ECORD"
60540  PRINT "3...FINE LAVORO"
60545  VTAB 4: HTAB 18: FLASH : PRINT
     "BY S.ORLANDO": NORMAL
60550  VTAB 18: INPUT "SELEZIONA
     PER NUMERO E <CR> ";SEL$
60555  TEXT
60560  ON  VAL (SEL$) GOSUB 30010
     ,32010,62010
60570  GOTO 60510
62000  REM FINE
62010  POP : TEXT : HOME : PRINT
     "BYE...": END
```

INIZIARE NEL MODO MIGLIORE

Guida al SINCLAIR ZX81 ZX80-Nuova ROM

Pagg. 262
Cod. 318B

L. 16.500



SCONTO 20% AGLI ABBONATI FINO AL 28-2-'84

IL LIBRO

Questa guida, con chiarezza, semplicità espositiva e ricchezza di esemplificazioni, risulta un vero e proprio strumento operativo per tutti coloro che vogliono avvicinarsi all'informatica in generale, e imparare la programmazione in BASIC, in particolare travalicando i tre calcolatori (ZX81, ZX80, ZX80 nuova ROM) a cui fa riferimento. Partendo da quello che è un computer, il lettore impara nei primi sei capitoli a programmare in BASIC, spingendosi, per chi lo vuole, oltre, sino alla programmazione in linguaggio macchina. L'ultimo capitolo riporta parecchi programmi e per ciascuno, vengono fornite, dove possibile, le diverse versioni. Tra l'altro si parlerà di file e di animazione delle figure. Per finire ben otto Appendici, essenziali ed utilissime, tra cui spiccano per interesse le due dedicate ai sistemi operativi dello ZX80, ZX80 nuova ROM e ZX81.

Per ordinare il volume utilizzare l'apposito tagliando inserito in fondo alla rivista.

SOMMARIO

Introduzione - Il calcolatore - Installazione del calcolatore - La programmazione - Il linguaggio BASIC - Come operare - Utilizzo della memoria - Linguaggio macchina - Esempi di programmi ---- caratteri del sistema - variabili del sistema - scheda BASIC ZX80 - scheda BASIC ZX80 nuova ROM e ZX81 - errori segnalati dalla macchina - sistema operativo dello ZX80 - sistema operativo dello ZX81 e nuova ROM

GRUPPO EDITORIALE JACKSON
Divisione Libri

PERSONAL SOFTWARE

Semplificazione di frazioni per TI 58-59

Il programma "tascabile" per conoscere rapidamente il M.C.D. ed il m.c.m. di due numeri e per semplificare le frazioni

di *Francesco Carbone*

Spesso, quando si lavora con espressioni matematiche, bisogna operare con frazioni anziché con numeri decimali. In tal caso è utile avere sul proprio calcolatore programmabile una semplice routine che permetta di conoscere in breve tempo il M.C.D. (Massimo Comune Divisore) ed il m.c.m. (minimo comune multiplo) di due numeri, e che provveda a ridurre ai minimi termini una frazione; inoltre la conoscenza del m.c.m. di due numeri interi si rivela utile nella somma di frazioni ed in molti altri calcoli matematici.

Il programma che segue si basa sull'Algoritmo di Euclide, è utile pertanto fare qualche precisione di carattere puramente matematico: dati due numeri interi a e b non entrambi nulli, un intero *d* si definisce massimo comune divisore di *a* e *b* se: 1) *d* è un divisore comune di a e di b; 2) se qualunque *c* che divide sia *a* che *b*, divide anche *d*. Ad esempio 2 è M.C.D. di 4 e − 6 in quanto i divisori comuni di 4 e − 6 sono 1, − 1, 2, − 2 e quindi per 2 vale la (1); inoltre 1, − 1, 2, − 2 sono tutti divisori di 2 e quindi per 2 vale anche la (2).

L'Algoritmo Euclideo è un procedimento di calcolo che permette, dati due numeri interi a e b, con b diverso da zero, di determinare il loro M.C.D.: si divide *a* per *b*, *b* per il resto trovato, il primo resto per il secondo resto, e così via fino ad ottenere resto zero.

L'ultimo resto diverso da zero è il nostro M.C.D..

Facciamo un esempio:

Vogliamo conoscere il M.C.D. di 22734 e 336.

$$22734 = 336 \times 67 + 222$$
$$336 = 222 \times 1 + 114$$
$$222 = 114 \times 1 + 108$$
$$114 = 108 \times 1 + 6$$
$$108 = 6 \times 18$$

L'ultimo resto diverso da zero è 6, quindi il M.C.D. di 22734 e 336 è 6.

Per quanto riguarda il minimo comune multiplo, la sua definizione è analoga a quella del M.C.D.: dati due numeri interi A e B un intero m si dice minimo comune multiplo di *a* e di *b* se: 1) m è un multiplo comune di a e di b; 2) ogni intero che è multiplo comune di a e di b, risulta anche multiplo di m.

È facile conoscere il m.c.m. di due numeri quando già si conosca il M.C.D.; infatti si dimostra che, restando invariato il segno, il prodotto di due interi a e b è uguale al prodotto del M.C.D. per il m.c.m., di a e di b.

Segue un breve programma che applica i principi. ■

000	76	Lbl	*025*	65	X	*050*	43	RCL
001	11	A	*026*	43	RCL	*051*	01	01
002	42	STO	*027*	01	01	*052*	95	=
003	00	00	*028*	95	=	*053*	91	R/S
004	42	STO	*029*	94	+/−	*054*	76	Lbl
005	02	02	*030*	85	+	*055*	15	E
006	91	R/S	*031*	43	RCL	*056*	43	RCL
007	76	Lbl	*032*	00	00	*057*	03	03
008	12	B	*033*	95	=	*058*	55	:
009	42	STO	*034*	67	x = t	*059*	43	RCL
010	01	01	*035*	00	0	*060*	01	01
011	42	STO	*036*	42	42	*061*	95	=
012	03	03	*037*	48	Exc	*062*	91	R/S
013	25	CLR	*038*	01	01	*063*	76	Lbl
014	32	x → t	*039*	42	STO	*064*	16	A'
015	91	R/S	*040*	00	00	*065*	43	RCL
016	76	Lbl	*041*	13	C	*066*	02	02
017	13	C	*042*	43	RCL	*067*	65	X
018	43	RCL	*043*	01	01	*068*	43	RCL
019	00	00	*044*	91	R/S	*069*	03	03
020	55	:	*045*	76	Lbl	*070*	55	:
021	43	RCL	*046*	14	D	*071*	43	RCL
022	01	01	*047*	43	RCL	*072*	01	01
023	95	=	*048*	02	02	*073*	95	=
024	59	int	*049*	55	:	*074*	91	R/S

ESEMPIO D'USO:

Vogliamo semplificare la frazione 6642/3510.
Impostiamo 6642 e premiamo A, impostiamo 3510 e premiamo B. Premendo C avremo il M.C.D. che è 54, premendo D avremo il numeratore della frazione (123) e premendo E, il denominatore (65). Il m.c.m. si ottiene premendo A'.

Figura 1. *Elenco delle istruzioni e codici di tasto.*

PERSONAL SOFTWARE

Semplificazione di frazioni per TI 58-59

Figura 2. *Il listato.*

```
000  76  LBL
001  11   A
002  42  STO
003  00   00
004  42  STO
005  02   02
006  91  R/S
007  76  LBL
008  12   B
009  42  STO
010  01   01
011  42  STO
012  03   03
013  25  CLR
014  32  X:T
015  91  R/S
016  76  LBL
017  13   C
018  43  RCL
```

Seguito figura 2.

```
019  00   00
020  55   ÷
021  43  RCL
022  01   01
023  95   =
024  59  INT
025  65   ×
026  43  RCL
027  01   01
028  95   =
029  94  +/-
030  85   +
031  43  RCL
032  00   00
033  95   =
034  67   EQ
035  00   00
036  42   42
037  48  EXC
038  01   01
039  42  STO
040  00   00
041  13   C
042  43  RCL
043  01   01
044  91  R/S
045  76  LBL
046  14   D
```

Seguito figura 2.

```
047  43  RCL
048  02   02
049  55   ÷
050  43  RCL
051  01   01
052  95   =
053  91  R/S
054  76  LBL
055  15   E
056  43  RCL
057  03   03
058  55   ÷
059  43  RCL
060  01   01
061  95   =
062  91  R/S
063  76  LBL
064  16   A'
065  43  RCL
066  02   02
067  65   ×
068  43  RCL
069  03   03
070  55   ÷
071  43  RCL
072  01   01
073  95   =
074  91  R/S
```

IL BASIC E LA GESTIONE DEI FILE

Il libro si rivolge in modo particolare a chi già conosce il Basic e desidera poter realizzare programmi che prevedano l'uso di file residenti su disco. Dopo aver preso in esame, utilizzando numerosi esempi pratici, le particolarità del Microsoft, si passa alla descrizione delle istruzioni necessarie ad una corretta gestione dei file su disco, sia ad accesso diretto che sequenziale. Una terza parte del libro è infine interamente dedicata alla esposizione dei metodi pratici per l'uso dei file ad accesso diretto e dei data base.

Cod. 515H **L. 11.000 Pagg. 164**

50 ESERCIZI IN BASIC

Una raccolta completa e progressiva di esercizi riguardanti matematica, gestione, ricerca operativa, gioco e statistica. Ciascun esercizio proposto comporta l'enunciazione e l'analisi del problema, la risoluzione mediante flow-chart e commenti, così come un programma che implementa la soluzione, illustrato da semplici esempi rappresentativi.
Questo metodo mette in grado il lettore di verificare passo passo le sue conoscenze e il livello di apprendimento raggiunto.

Cod. 521A **L. 13.000 Pagg. 208**

75 PROGRAMMI IN BASIC PER IL VOSTRO COMPUTER

Il volume raccoglie ben 75 programmi originali scritti in un Basic generico, utilizzabili, salvo poche e semplici modifiche, sulla maggior parte dei personal computer in commercio, a cassetta come a floppy disk. Per ciascuno, dopo una descrizione introduttiva, viene fornito il listing e un campione di esecuzione. Così come sono, i programmi proposti (tutti verificati) costituiscono un valido ausilio per chiunque debba risolvere problemi di matematica, statistica, finanza o genericamente di pratica utilità.

Cod. 551D **L. 12.000 Pagg. 196**

GIOCARE IN BASIC

Nei giochi, il lettore può ritrovare tutte quelle situazioni reali di programmazione che gli saranno indispensabili nella comprensione e realizzazione di qualsiasi applicazione interattiva del proprio computer, anche le più sofisticate. Questo senza annoiarsi, ma entrando da subito all'interno della materia per imparare a comprendere il BASIC, il proprio computer e i computer in genere.

Cod. 522A **L. 20.000 Pagg. 324**

... dalla libreria JACKSON

DIVISIONE LIBRI



SCONTO 20% AGLI ABBONATI FINO AL 28-2-'84

CEDOLA DI COMMISSIONE LIBRARIA

Ritagliare (o fotocopiare) e inviare a Gruppo Editoriale Jackson Via Rosellini, 12 - 20124 Milano

Nome e Cognome ______________________

____

Indirizzo ______________________

____

Cap. ________ Città ______________________

Provincia ______________

Codice Fiscale (indispensabile per le aziende)

Inviatemi i seguenti libri:

Codice Libro	Quantità	Codice Libro	Quantità	Codice Libro	Quantità	Codice Libro	Quantità

□ Pagherò al postino il prezzo indicato + l. 2.000 per contributo fisso spese di spedizione

□ Allego assegno n° .... di L. ....

Data .... Firma ....

Memorie associative e ordinamento

Interessante illustrazione di due algoritmi traducibili in qualunque linguaggio

di *Maurizio Coccorese*

	CAMPO 1	CAMPO 2
1	LUPO	010101011
2	TOPO	11111111
3	LUPO	00000001
⋮	⋮	⋮

LUPO	??????

Dovendo ricercare LUPO, quale dobbiamo considerare?

Figura 1. *In questo caso il computer non saprebbe quale "entrata" considerare e quale valore del campo 2 prendere.*

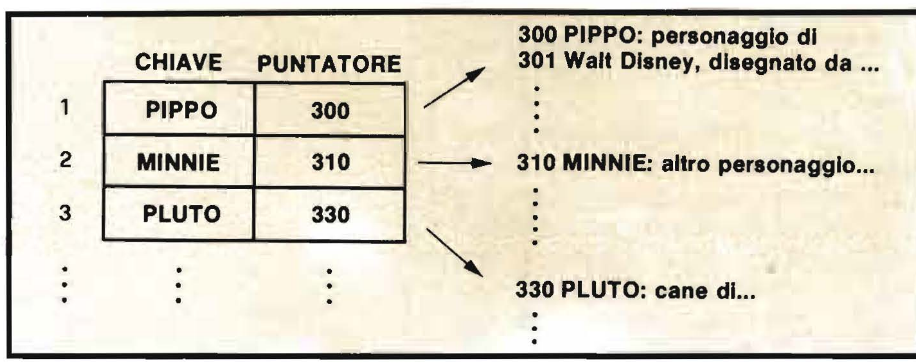


Figura 2. *Ad ogni chiave è associato il puntatore ad un altra tabella, che contiene le informazioni richieste. Ad esempio le informazioni della chiave MINNIE (linea 2) sono contenute nelle linee da 310 in poi della seconda tabella.*

Qualche parola d'introduzione

Nel numero 5 di **Personal Software** si è visto come ricercare un elemento in tablle per mezzo di ricerche lineari, binarie ed hash. Lo scopo di questo articolo è quello di centrare l'attenzione sul significato di memoria associativa realizzata tramite quelle stesse tabelle e quindi di sviluppare due algoritmi per l'ordinamento necessario in una tabella dicotomica. La mia intenzione non è quella di preparare due programmi in un linguaggio di programmazione specifico, bensì quella di fornire due algoritmi, interpretabili in qualsiasi linguaggio di programmazione, che diano al lettore una idea della soluzione del problema dell'ordinamento. Infatti di algoritmi di ordinamento ce ne sono molti, più o meno complicati, e tra i tanti ne sono stati scelti due, tra cui l'algoritmo di Shell, caratterizzato da aspetti matematici interessanti.

Inizialmente viene illustrato il significato di memoria associativa, quindi vengono richiamati i concetti della ricerca di tipo binario e infine vengono approfonditi i due algoritmi di ordinamento, dei quali si forniscono anche i diagrammi di flusso.

Memoria associativa

Genericamente si può pensare a una memoria di tipo associativo come a una struttura che è in grado di associare, la parola stessa lo dice, dei determinati valori alfanumerici a particolari insiemi di valori o dati; nulla vieta di pensare che questi particolari insiemi siano anche vuoti, cioè non contengono alcun elemento o più precisamente alcun dato. Possiamo realizzare una memoria di tipo associativo molto semplice, creando delle tabelle formate da elementi detti "entrate", ognuna delle quali consistente in due campi: nel primo rappresentiamo un determinato valore alfanumerico detto CHIAVE che ci permette di accedere univocamente ad un determinato valore ad esso associato contenuto nel secondo campo. È evidente che per ogni valore la chiave deve essere unica; infatti se la chiave è in grado di aprire più serrature con sequenzialmente è in grado di ottenere più valori generando delle ambiguità che un computer non istruito opportunamente non sarebbe in grado di risolvere (figura 1).

Idealmente nel cervello umano accade la stessa cosa, con meccanismi certo differenti, quando raggruppiamo, per poterle meglio ricordare, una serie d'informazioni sotto un acronimo. Per esempio invece di ricordare American Standard Code of Interchange Information è molto più comodo ricordare il

Memorie associative e ordinamento

relativo acronimo cioè ASCII, dove ASCII è la chiave di una zona recondita del cervello che contiene un'informazione che sarebbe più difficile da ricordare immediatamente. Per il cervello elettronico il vantaggio di una memoria associativa non è quello di ricordare meglio ma è quello di velocizzare le ricerche per contenuto, creando un'elasticità maggiore nell'organizzazione delle liste d'informazioni che possono essere smembrate ed esaminate separatamente. Infatti nel secondo campo di ogni "entrata" si può porre, invece che l'informazione desiderata, un "puntatore" ad un'altra tabella che contiene effettivamente le informazioni; basterà cambiare questo puntatore per cambiare serie d'informazioni arbitrariamente grandi (figura 2).

Inoltre, dal momento che le chiavi associano univocamente determinate informazioni, basterà in ogni istante far riferimento ad esse e non all'intera serie d'informazioni, evitando continue estrazioni di dati dalla memoria. Individuare un gruppo d'informazioni vorrà dire semplicemente individuare una chiave. È noto che gli attuali computer hanno tempi di elaborazione rapidissimi, tuttavia la memoria rappresenta il classico collo di bottiglia in grado di rallentare il flusso delle informazioni in un rapporto di 1 : 10 in situazioni non rare. Memorie di tipo associativo permettono di superare i limiti di velocità imposti dalle memorie principali RAM che la tecnologia moderna non è riuscita ancora a superare. Infatti se moltiplichiamo il tempo piuttosto basso necessario per estrarre una parola dalla memoria di un ipotetico computer, confrontarla con un'altra già estratta e riporre il risultato nuovamente in memoria, per le migliaia di parole che normalmente vengono trattate, si può immaginare quanto questo problema sia importante.

Inoltre ci sono vantaggi per l'operatore, al quale le chiavi possono suggerire mnemonicamente i contenuti dei rispettivi campi informativi.

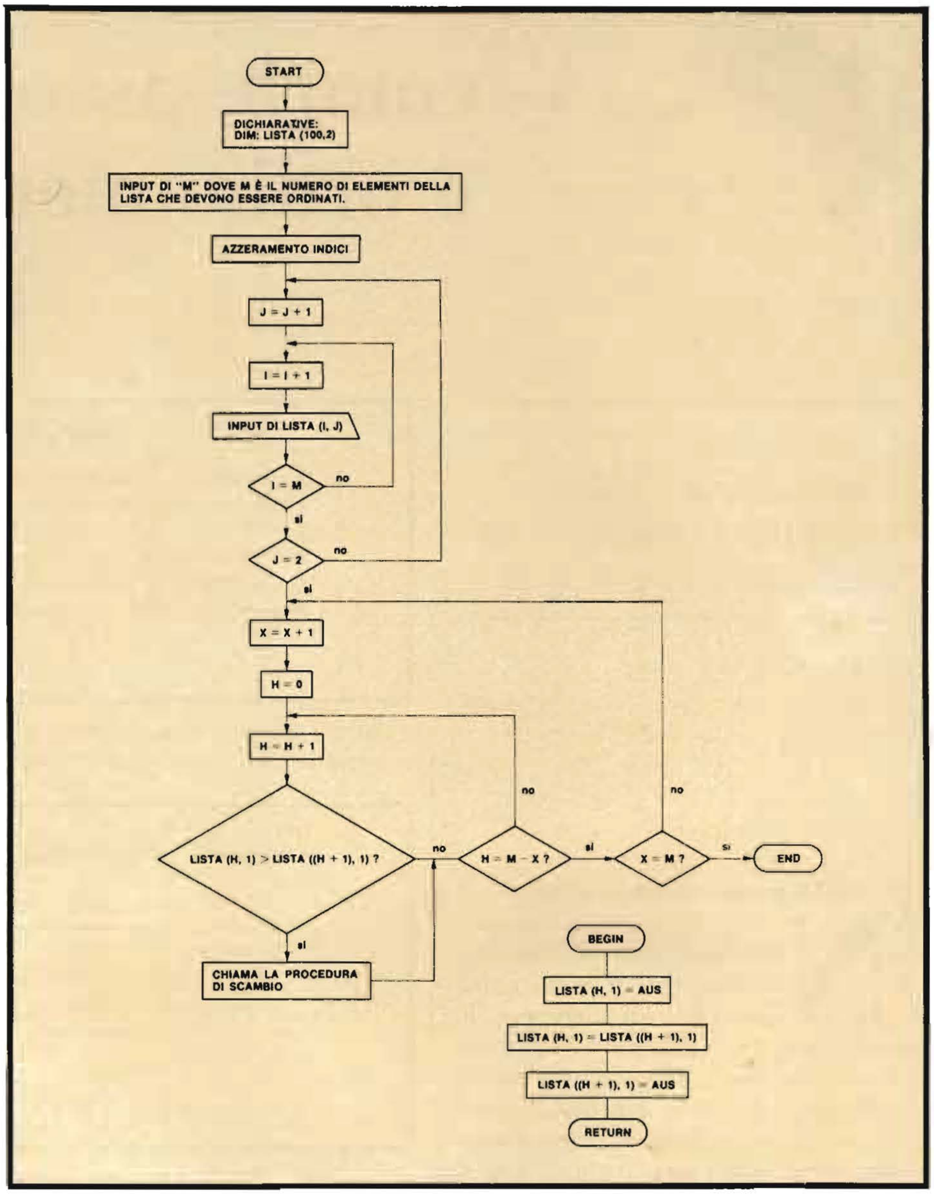


Figura 3. *Questo diagramma di flusso rappresenta il primo algoritmo di ordinamento su una lista formata da 100 elementi al massimo, ognuno dei quali costituito da due campi. Vengono caricate prima le chiavi e poi la parte informativa. Sotto è rappresentata una procedura di scambio.*

Basti pensare alla funzione del titolo di un argomento che ci suggerisce, a grandi linee, il possibile contenuto. Comunque, per quanto una memoria associativa può velocizzare una ricerca, resta sempre il fatto che una lista può assumere proporzioni tali da richiedere particolari metodi di ricerca che limitino i confronti al numero strettamente necessario.

Ricerca di tipo binario o dicotomico

Una ricerca dicotomica o semplicemente binaria, consiste nel dividere una lista ordinata ad esempio in modo crescente in due parti uguali finchè non si trova l'elemento ricercato. Effettivamente la ricerca avviene così: si considera l'elemento di centro della lista e lo si definisce "chiave di mezzo", dopo di che si confronta tale chiave con quella che si sta cercando; se sono uguali la ricerca si interrompe altrimenti si stabilisce quali delle due è maggiore; se la chiave di mezzo è minore si ripete il processo per quella parte della lista che è al di sotto di essa poichè, la chiave da ricercare sarà senza dubbio nella seconda metà della lista dato che essa è ordinata; se viceversa la chiave di mezzo è maggiore si scarta la metà inferiore della lista. Tale processo va avanti finchè o non è più possibile dividere

in due la lista (e quindi l'elemento ricercato non viene trovato), oppure l'esito di un confronto è positivo.

In una tabella di 2^n elementi saranno necessari al massimo n confronti. Se da una parte questo metodo ci permette di ridurre notevolmente il numero di confronti da fare rispetto un tipo di ricerca lineare, presuppone, dall'altra, che la lista sia ordinata.

Primo algoritmo di ordinamento

Vediamo un semplice metodo di ordinamento il cui diagramma di flusso è dato in figura 3. Cerchiamo di capire come funziona; come prima operazione si considerino i primi due elementi della lista; supposto di avere stabilito un ordine crescente, si confrontano questi elementi per vedere se sono in ordine corretto; se non lo sono si scambiano cioè si permutano i loro posti.

Quindi si ripete l'operazione tra il secondo e il terzo elemento della lista, tra il terzo e il quarto fino all'(n-1)-esimo e l'n-esimo. Il risultato di questo primo "*giro*" è quello di avere l'elemento più grande al fondo della lista. Se gli elementi della lista sono n si deve ripetere questo processo n-1 volte e si avrà come risultato finale una lista di n elementi ordinati progressivamente; tuttavia dal momento che il primo giro mette l'elemento più grande nel posto più basso, il secondo mette il secondo elemento più grande nel penultimo posto e così via il terzo e il quarto fino all'n-esimo, possiamo dire che ogni giro ha un proprio massimo ad esso relativo che va ad assumere il posto più basso per quella tornata. Quindi per il secondo giro non c'è bisogno di fare il confronto fra l'ultimo elemento e il penultimo poichè nell'ultimo posto c'è già il massimo del giro precedente (per il primo giro soltanto il massimo assoluto). La stessa situazione si presenta per ogni coppia di valori successivi, tanto nell'ultima tornata ((n-1)-esima) si effettuerà il confronto unicamente

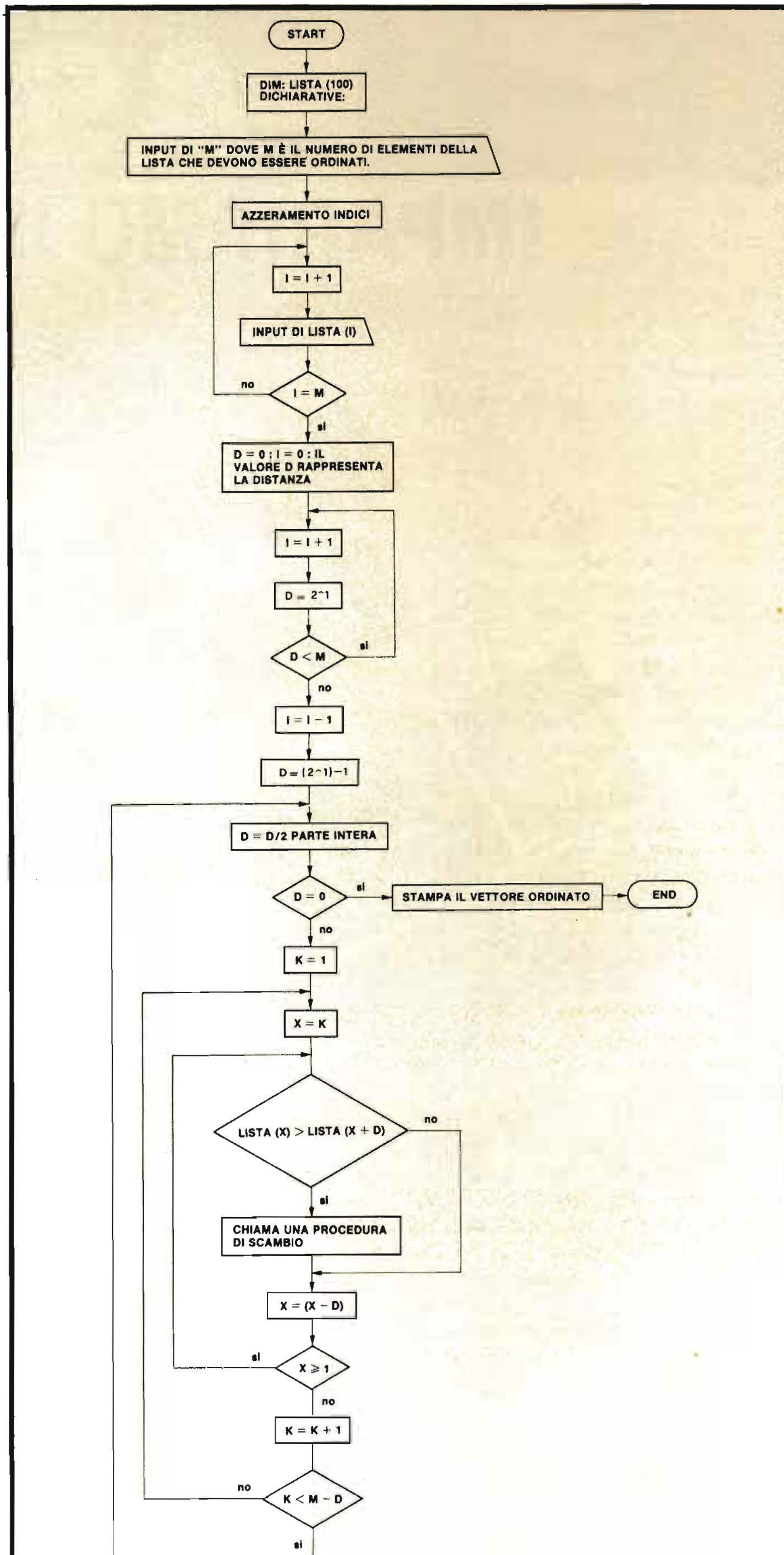


Figura 4. *Questo è l'algoritmo di Shell applicato ad una lista di 100 elementi al massimo. La prima parte stabilisce la distanza e la seconda il flusso di ordinamento. (D parte intera di 0,5 = 0).*

PASCAL

IMPARIAMO IL PASCAL



Pagine 162 Prezzo Lit. 11.500

Compattezza, concisione, chiarezza e notevoli potenzialità scientifiche, oltre a prestarsi ottimamente per calcoli gestionali e ad essere usato anche con i microcomputer, sono le caratteristiche che decretano il successo del PASCAL come linguaggio di programmazione. Non vi era però finora un testo che insegnasse a tutti a programmare in PASCAL: o perchè i libri esistenti sono troppo concisi, o troppo semplici, oppure perchè richiedono la conoscenza di altri linguaggi di programmazione, o, non ultimo, perchè in inglese. Queste sono proprio le lacune che colma il libro un libro di divulgazione, incentrato sull'auto-apprendimento, che non tedia con accademismi non funzionali il lettore riportandolo "a scuola". I capitoli sono il più possibile organici, in modo che la loro consultazione sia semplice ed agevole. Un riassunto di quanto si apprenderà è posto all'inizio e non in fondo al capitolo, perchè il lettore possa subito avere un metro valutativo con cui verificare passo-passo il suo apprendimento. E poi, ci sono consigli, problemi, esercizi affinchè il libro sia "usato" e non letto, perchè occorre sapere come si usa un'istruzione piuttosto che conoscerne le differenze semantiche tra linguaggio e linguaggio. Con un lavoro graduale, partendo senza alcuna conoscenza di programmazione, dopo circa due settimane dovreste conoscere abbastanza bene il PASCAL. Un buon risultato, no?!

SCONTO 20% AGLI ABBONATI
FINO AL 28-2-'84

Per ordinare il volume utilizzate l'apposito tagliando d'ordine inserito in fondo alla rivista.

Formato 15 x 21

Codice 501A

SOMMARIO

0. Da non trascurare
1. Come si descrive la sintassi del linguaggio
2. Come si scrive in PASCAL
3. Il programma e le dichiarazioni in generale
4. Le dichiarazioni ed i tipi standard
5. I tipi speciali e subrange
6. Gli statements di assegnazione
7. Gli statements di ripetizione
8. Gli statements logici
9. I dati strutturati - Generalità
10. Il tipo array
11. Il tipo record
12. Il tipo set
13. Il tipo file
14. Il tipo pointer
15. Le procedure e le funzioni
16. Procedure ricorrenti input ed output
17. I diagrammi di struttura

GRUPPO EDITORIALE JACKSON **DIVISIONE LIBRI**

tra il primo e il secondo elemento della lista. Come risultato si avrà una lista di elementi ordinata in modo crescente.

Algoritmo di Schell

Tutti gli studenti d'informatica prima o poi devono studiare l'algoritmo di Shell che risulta essere difficile da comprendere perchè, per capirlo bene bisogna farne il trace, cosa che tutti gli studenti trovano antipatica. Tuttavia questo algoritmo è interessante se non altro per i suoi aspetti di tipo matematico dal momento che come velocità di esecuzione non è inferiore al precedente. Concettualmente si tratta di muovere gli elementi più grandi della lista verso il fondo e gli elementi più piccoli verso l'alto, senza assegnare ad essi un posto stabile se non alla fine del processo. I confronti si fanno sempre fra coppie che questa volta non sono più consecutive ma caratterizzate da una distanza sempre decrescente. Il primo passo dell'algoritmo consiste nello stabilire la distanza iniziale che deve dividere le prime coppie. Si stabilisce la prima potenza di due al di sotto della dimensione della lista: se per esempio gli elementi sono 1000 la prima potenza di due al di sotto di questo numero è 512; se gli elementi sono 10 la prima potenza di due è 8.

Decrementiamo ora questa potenza di 1 e il risultato si divide per due considerando la parte intera del quoziente. Questo valore sarà la prima distanza che dividerà i primi confronti. Detta d questa distanza ed n la dimensione della lista i confronti si ripeteranno n-d volte. La seconda distanza sarà la metà della prima, la terza la metà della seconda e così via. Però per una stessa distanza gli elementi vengono "pettinati" più volte come mostrato nel diagramma a blocchi della figura 4. Il processo infine non si ripete più quando la distanza è uguale a uno. Il risultato è una lista ordinata progressivamente, quando la distanza diventa nulla.

Visti questi algoritmi dobbiamo tener presente che essi non sono gli unici, ci sono anche altri più complicati e più veloci o più facili o meno veloci. In generale possiamo dire che non esistono algoritmi che si adattino universalmente a tutte le necessità (situazione sempe presente in informatica); sta a noi scegliere quello che più si adatta alle nostre esigenze. L'ultima parola, quindi rimane sempre all'uomo e alla sua inventiva. ■

PROGRAMMI DI MATEMATICA E STATISTICA

Leggendo questo libro il lettore potrà formarsi quella logica di base indispensabile per la risoluzione di problemi di matematica e statistica.
Ad ogni programma viene preposta un'esposizione schematica del metodo numerico e delle tecniche di programmazione utilizzate, il diagramma a blocchi relativo all'algoritmo, il listato (anch'esso ottenuto da calcolatore) in cui tra l'altro vengono specificati il tempo e la quantità di memoria impiegati.

Cod. 522D **L. 16.000 Pagg. 228**

INTRODUZIONE AL PASCAL

Il volume, incentrato su numerosissimi esempi che verificano costantemente l'apprendimento del lettore, insegna a conoscere, capire ed usare tutte le particolarità e i vantaggi di questo linguaggio. Nel corso della trattazione vengono ampiamente utilizzate le tecniche di programmazione strutturata, come pure tecniche particolari, quali il trattamento dei file, l'utilizzazione della recursività e il trattamento grafico.

Cod. 516A **L. 30.000 Pagg. 484**

COMPUTER GRAFICA

Si può dire che la computer grafica si pone nel contesto più generale del trattamento dell'informazione, avendo individuato nell'immagine un contenuto informativo che è possibile elaborare.
Quest'opera, con il suo rigore informativo e scientifico, si pone come fondamentale nel carente panorama italiano; inoltre le informazioni e gli spunti contenuti nel testo contribuiranno certamente alla divulgazione ed alla formazione di idee nuove e feconde.

Cod. 519P **L. 29.000 Pagg. 174**

APPLE II - Guida all'uso

Se possedete un Apple e volete conoscerlo a fondo, se volete comprarlo, o se semplicemente volete imparare la sua programmazione, troverete in questo libro, tutte le risposte, comprese alcune vere "primizie" che vi occorrono per una perfetta operatività del sistema. Conoscerete i vari componenti del sistema e come usarli al meglio. Verrete guidati alla programmazione in BASIC e a usare le caratteristiche grafiche e sonore del sistema. Imparerete a memorizzare su disco sia programmi che archivi dati, come ad inserire un programma scritto in assembler in uno scritto in BASIC.
E poi ancora, tutte le istruzioni e funzioni BASIC e ben 12 appendici veramente basilari.

Cod. 331P **L. 26.000 Pagg. 400**

CEDOLA DI COMMISSIONE LIBRARIA

Ritagliare (o fotocopiare) e inviare a
Gruppo Editoriale Jackson Via Rosellini, 12 - 20124 Milano

Nome e Cognome ______________________________

____

Indirizzo ______________________________

____

Cap. __________ Città ____________________ Provincia __________

Partita I.V.A. (indispensabile per le aziende)

Si richiede l'emissione della fattura □

Inviatemi i seguenti libri:

Codice Libro	Quantità	Codice Libro	Quantità	Codice Libro	Quantità	Codice Libro	Quantità

□ Pagherò al postino il prezzo indicato + L 2.000 per contributo fisso spese di spedizione

□ Allego assegno n° .... di L. ....

Data .... Firma ....

□ Non Abbonato □ Abbonato sconto 10% □ L'Elettronica □ Elettronica Oggi □ Automazione Oggi □ Elektor □ Informatica Oggi □ Computerworld □ Bit □ Personal Software □ Strumenti Musicali □ Videogiochi

... dalla libreria JACKSON



SCONTO 20% AGLI ABBONATI FINO AL 28-2-'84

GRUPPO EDITORIALE JACKSON

Divisione Libri

Grafici ad alta risoluzione con il TI99/4A

Un programma BASIC che vi permette di visualizzare qualunque funzione matematica

di *Sergio Borsani*

È proprio vero che l'erba del vicino è sempre più verde! Penso al disagio che prova qualcuno nel constatare che il proprio computer non è in grado di offrire alcune prestazioni desiderate. So di qualcuno che pensava di poter ottenere con un TI-99 grafici tridimensionali sul tipo di quelli rivolti alla progettazione industriale. Scagli la prima pietra chi non ha sperato di poter ottenere almeno il grafico ad alta risoluzione di una funzione matematica. La difficoltà consiste nel fatto che in TI BASIC non è possibile indirizzare indipendentemente i punti video (pixel). Tuttavia si può, grazie al nuovo video processor TMS9918A che il TI99/4A possiede al posto del TMS9918 del precedente modello, definire un carattere punto per punto e successivamente stamparlo nella posizione opportuna.

Con l'home computer, un programma BASIC che abbia la pretesa di costruire un grafico ad alta risoluzione deve innanzitutto "scandire" il video da sinistra a destra pixel per pixel e calcolare, in corrispondenza delle 256 colonne, il valore della funzione che si vuole rappresentare.

Per distinguere le 24 righe e le 32 colonne disponibili nel modo grafico dalle 192 righe e 256 colonne disponibili nel modo bit-mat, potremmo convenire di chiamare queste ultime pix-righe e pix-colonne.



Al termine della prima operazione il computer avrà memorizzato, per ogni pix-colonna, il numero di pix-riga corrispondente al valore della funzione. Dovrà poi colmare tutte le lacune eventualmente esistenti tra ogni punto ed il successivo, mettendo in ON tutti i pixel tra essi compresi.

Ogniqualvolta si verifica il salto di riga o di colonna passerà quindi a definire la stringa esadecimale atta a specificare il carattere particolare che rappresenta un piccolo tratto della curva considerata e a stamparlo nella posizione opportuna con l'istruzione CALL HCHAR.

Questo per quanto riguarda i concetti di base.

Il programma che presento nel listato 1 ovviamente è strutturato in conformità a quanto ho premesso e in più possiede delle prerogative che saranno sicuramente molto apprezzate dai lettori. Per citare le più importanti, una consiste nel dimensionamento automatico del grafico, un'altra nell'uso di una routine per la "somma" dei caratteri in modo che la curva non cancelli parzialmente gli assi di riferimento quando li interseca o anche quando passa in prossimità di essi.

Quest'ultima soluzione semplice ed originale è il naturale completamento di un programma rivolto alla grafica ad alta risoluzione e merita una breve descrizione.

Sommiamo i caratteri

Se si stampa un carattere sul video esso sostituisce ogni altro simbolo eventualmente presente nella stessa posizione; alla regola non sfuggono nemmeno i caratteri speciali.

Se un tratto di curva passa in prossimità di un asse cartesiano o addirittura lo interseca, i caratteri che lo rappresentano andranno a sostituire quelli che rappresentavano l'asse creando dei vuoti a dir poco antiestetici. La "somma" dei caratteri consiste nel definire, quando è necessario, nuovi caratteri che portano in ON tutti i pixel del carattere che rappresenta un tratto di curva e di quello che invece rappresenta una parte dell'asse di riferimento.

Per far questo le stringhe esadecimali che definiscono i caratteri vengono decodificati ed i loro simboli vengono sommati ad uno ad uno in base ad una tabella di composizione memorizzata nel programma (istruzioni DATA delle linee 300-440).

Ogni simbolo esadecimale controlla un blocco di 4 pixel; F si riferisce ad un blocco con tutti e quattro i pixel in posizione ON. È evidente

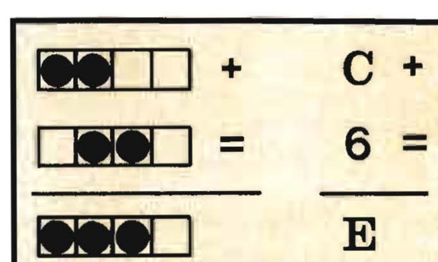


Figura 1. *È possibile sommare due caratteri sommando uno ad uno gli elementi delle stringhe esadecimali che li definiscono in base ad una tabella di composizione.*

Grafici ad alta risoluzione con il TI99/4A

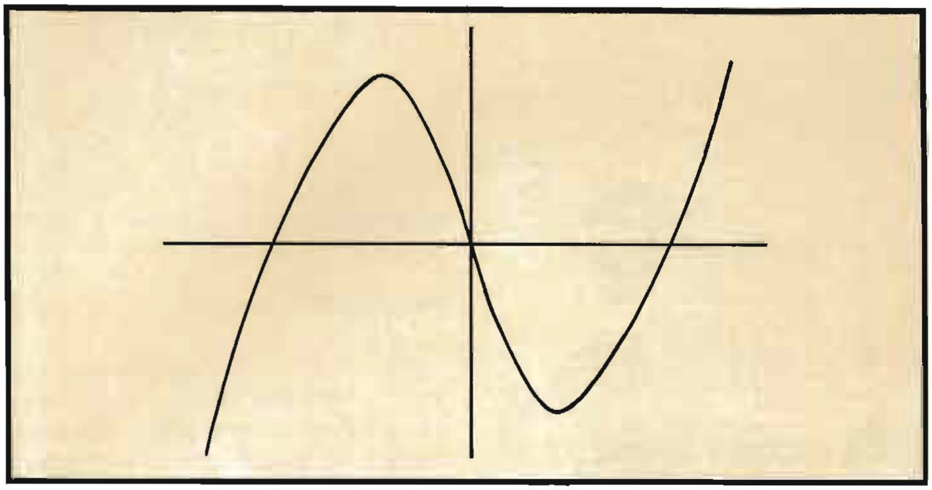


Figura 2. *A titolo dimostrativo nel programma è memorizzata la funzione* $y = x^3 - 3x$. *È qui riprodotta l'immagine elaborata dal computer.*

che F sommato a qualsiasi altro simbolo esadecimale dà ancora F; per la costanza dei risultati questo elemento non è stato inserito nella tabella.

Consideriamo ad esempio la somma C + 6, il risultato è E; infatti C è un blocco con il primo ed il secondo pixel ON ed il terzo ed il quarto OFF, 6 è un blocco con il secondo ed il terzo pixel ON; la loro somma è un blocco con il primo, il secondo ed il terzo pixel ON ed il quarto OFF, esso corrisponde al simbolo esadecimale E.

La figura 1 illustra questo concetto di somma.

Utilizziamo il programma

Prima di esaminare dettagliatamente il listato vediamo come si utilizza il programma. Un messaggio ricorda che la funzione della quale si vuole ottenere il grafico va definita alla riga 850. Chi non l'avesse fatto prima del RUN deve fermare il programma con il tasto FCTN(4), richiamare la linea 850, dove troverà già una funzione definita, sostituirla con la propria e lanciare nuovamente il programma. Proseguendo, bisogna specificare l'intervallo nel quale si studia la funzione, X1 è l'estremo inferiore e X2 è l'estremo superiore.

Il grafico è dimensionato automaticamente nel senso che sia la posizione degli assi sia l'unità di misura vengono calcolate per ottenere la migliore rappresentazione, in modo da utilizzare tutti i 192 pixel sull'asse verticale evitando che una parte della curva debba cadere fuori dallo schermo.

L'unica attenzione da parte dell'utente deve essere volta ad escludere dall'intervallo valori per i quali non può essere calcolata la funzione.

Ad esempio la funzione Y = SQR (1 − X x X) è definita solo nell'intervallo − 1 + 1; per valori diversi il radicando diventa negativo, appare il messaggio BAD ARGUMENT ed il programma si blocca. Se la funzione è Y = 36/X, l'intervallo non può comprendere il valore 0 perchè non esiste il valore della funzione in quel punto; ad esempio, si potrà ottenere il grafico nell'intervallo + 1 + 36. In questo caso l'asse delle Y non appare sullo schermo ed anche il grafico appare tutto spostato sulla sinistra. Prevedendo tali situazioni, il programma è stato dotato di una routine che consente il cambiamento dell'origine degli assi cartesiani, controllato dai quattro tasti muniti di frecce.

Riferendoci ancora alla funzione Y = 36/X, nell'intervallo + 1 + 36, premendo un tasto al termine del dimensionamento appare il solo asse orizzontale nell'ultima riga in basso, l'asse delle Y è virtualmente spostato a sinistra fuori dallo schermo.

Premendo due o tre volte il tasto (D) con la freccia rivolta a destra è possibile ottenere la stampa anche dell'asse Y.

È opportuno non abusare di questa funzione e spostare gli assi solo del minimo necessario perchè così si sposta anche tutta la stampa del grafico ed una parte di questo potrà non apparire sul video.

La figura 2 mostra il grafico della funzione Y = X^ 3−3 x X, attualmente presente nel programma, nell'intervallo − 2 + 2, senza traslazione degli assi.

Il computer impiega poco più di due minuti per il dimensionamento automatico e circa quattro minuti per la stampa del grafico con l'utilizzazione di 74 caratteri definiti da programma.

Un tuffo nel listato

Alla riga 230 vengono dimensionate le matrici; S$ contiene la tabella di composizione per la somma dei caratteri, P è il numero di pixcolonna dei punti del diagramma calcolati per valori discreti, ESA$ memorizza gli elementi per comporre la stringa esadecimale che definisce i caratteri grafici, infine E$ contiene in sequenza i simboli esadecimali.

Delle istruzioni DATA si è già parlato in precedenza, esse contengono la tabella di composizione per la somma dei caratteri.

Alla linea 680 e seguente i dati in input: X1 ed X2 sono gli estremi dell'intervallo nel quale viene studiata la funzione. Alla riga 850 c'è la funzione della quale si vuole ottenere il grafico.

Le istruzioni 920-1070 ricavano il valore unitario corrispondente ad un pixel; esso è posto inizialmente

uguale alla duecentoquarantesima parte dell'intervallo X2-X1, se per tale valore il massimo relativo (MAX) o il minimo relativo (MIN) cadono fuori dallo schermo, allora il valore unitario (U) è posto uguale alla centosettantaseiesima parte dell'intervallo MAX-MIN. In tal modo tutta la funzione, da Y = f (X1) a Y = f (X2), viene rappresentata sullo schermo.

RIGA 0 e COLO 0 (righe 1110-1120) sono le coordinate dell'origine mentre X0 e Y0 sono le stesse espresse in numero di pix-colonna e pix-riga.

Il ciclo FOR NEXT (righe 1160-1280) carica i valori della matrice unidimensionale P (vedi figura 3).

INIZIO (1250) e FINE (1290) indicano proprio l'inizio e la fine della stampa, in numero di pix-colonna.

Alla riga 1340 viene mandato l'ultimo messaggio, in seguito saranno ridefiniti tutti i caratteri ASCII che non saranno più disponibili per il testo.

Segue la stampa degli assi cartesiani, TRSX0 e TRSY0 controllano la traslazione degli assi, il loro valore è determinato nel sottoprogramma 2920.

Con il ciclo FOR NEXT 1820 inizia la stampa del grafico.

Le principali variabili utilizzate sono P1, numero di pix-riga del primo punto, RIGA 1 e COLO 1, numeri di riga e colonna, NRC, numero di pix-riga del pixel all'interno del carattere, NCC, numero di pix-colonna del punto all'interno del carattere (un carattere è formato da 8 pix-righe e 8 pix-colonne), NB1, numero del blocco nel quale si trova il pixel (un carattere è formato da 16 blocchi di 4 pixel ciascuno), ESADEC1 è il corrispondente valore decimale del simbolo esadecimale al quale si risale con la funzione E$ (ESADEC1). Al punto seguente, P2, sono associate le stesse variabili con indice 2.

Al termine di ogni ciclo il vecchio punto conseguente diventa il nuovo antecedente ed un nuovo punto diventa il nuovo conseguente (istruzioni 2270-2300).

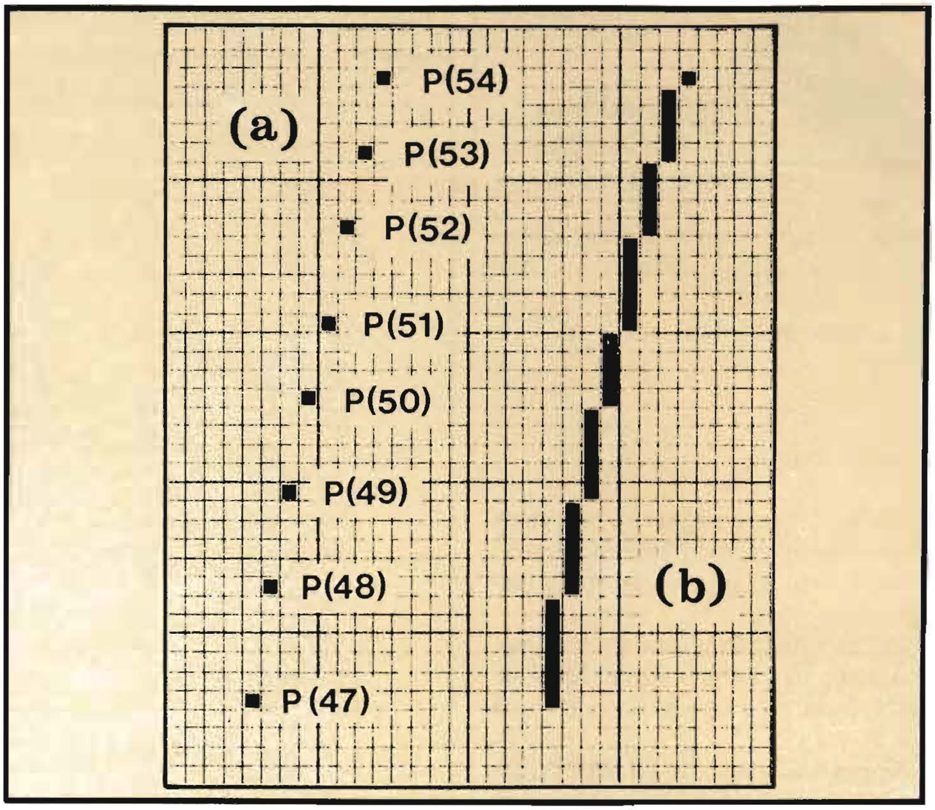


Figura 3. *Nella fase di dimensionamento si calcola la curva per punti (a). Successivamente vengono colmati i vuoti tra punto e punto e definiti i caratteri per la stampa. La mappa-video (b) mostra, ingrandito, un breve tratto della curva in esame.*

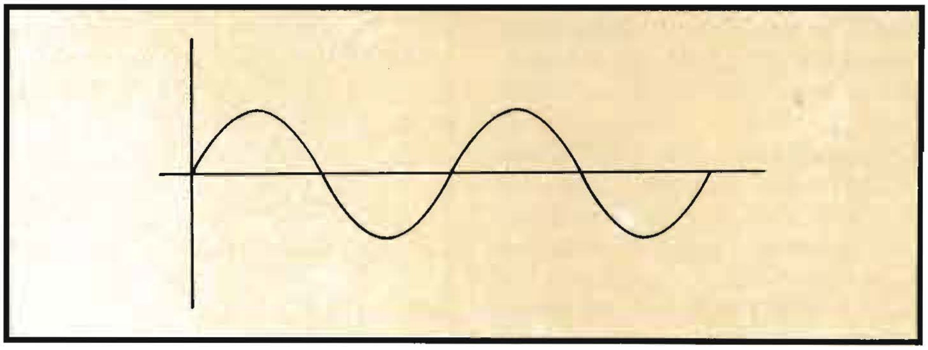


Figura 4. *Dimensionamento automatico e traslazione degli assi consentono sempre la migliore rappresentazione grafica. La funzione è Y = SIN(X).*

Se tra due punti P1 e P2 la differenza tra i numeri di pix-riga è maggiore di 1 il ciclo FOR NEXT 2000-2090 si incarica di stampare punti sulla stessa pix-colonna di P1 fino a colmare il vuoto esistente tra i due punti (la differenza tra i numeri di pix-colonna, invece, è sempre uguale a 1).

Anche durante questo ciclo se cambia il numero di riga o di colonna il carattere deve essere stampato.

Quando P1 e P2 si trovano sulla stessa pix-riga il programma controlla se si trovano sullo stesso blocco tra i 16 che formano un carattere perchè se ciò si verifica va calcolato il nuovo valore esadecimale da associare al blocco (linea 2240).

Nell'ultima parte del listato le REM rendono esplicito il contenuto delle quattro routine utilizzate dal programma.

La funzione CALL GCHAR (linea 2500) indaga se lo spazio destinato alla stampa di un carattere è già occupato, nel qual caso si procede alla somma dei caratteri.

La stampa di un carattere è sempre seguita dall'azzeramento di

PERSONAL SOFTWARE

Grafici ad alta risoluzione con il TI99/4A

ESA$ che contiene gli elementi della stringa esadecimale che lo hanno definito.

La memoria richiesta è di 7.1 Kbyte.

Ultimi consigli

Qualche breve suggerimento permetterà di utilizzare al meglio il programma e di acquisire la pratica necessaria.

1) Lasciate la funzione attualmente scritta nel programma. Date il RUN. Indicate l'intervallo X 1 = − 2 e X 2 = 2.

Apparirà la scritta DIMENSIONAMENTO RUNNING. Dopo poco più di due minuti appaiono alcuni messaggi. Premendo un tasto qualsiasi si ottiene la stampa degli assi cartesiani con l'origine al centro del teleschermo. Premendo il tasto SPACE o qualsiasi altro tasto che non sia uno di quelli con le frecce, inizierà la stampa del grafico (vedi figura 2).

2) Modificare la funzione alla riga 850 del programma con la seguente Y = 18/X. Date il RUN. Poichè per X = 0 non esiste il valore della funzione, specificate il seguente intervallo: X 1 = 1 e X 2 = 18. Non è opportuno attribuire a X 1 valori prossimi allo zero perchè per X tendente a zero il valore della funzione sale rapidamente e, per rappresentare tutto il tratto di curva, viene notevolmente ridotta, in proporzione, la rappresentazione dell'intervallo di definizione.

Dopo l'ultimo messaggio, premendo un tasto non appare nulla sullo schermo: l'origine degli assi cade al di fuori. Si potrebbe ottenere ugualmente il grafico della funzione: con la stampa degli assi esso è tuttavia più completo.

Premete allora il tasto E (↑), apparirà l'asse orizzontale, poi tre volte il tasto D (→) e apparirà anche l'asse verticale. Premete un altro tasto qualsiasi e verrà stampato il grafico.

3) Inserite la funzione Y = 3 x X + 4SQR (1 − X x X). Abbiamo già detto che il radicando non può essere negativo, pertanto X non può essere minore di − 1 nè maggiore di + 1. Indicate l'intervallo X 1 = − 1 e X 2 = 1. Procedete come in prima.

Non è necessaria la traslazione degli assi perciò dopo la loro stampa premete un tasto qualsiasi eccettuati quelli con le frecce.

4) Inserite la funzione Y = SIN(X) ed indicate l'intervallo X 1 = 0 e X 2 = 12.6. I valori di X sono espressi in radianti e 12.6 rappresenta circa 4 π. L'origine degli assi apparirà spostata a sinistra piuttosto in alto. Il computer ha calcolato esattamente lo spazio necessario per la stampa del grafico, tuttavia questa avverrà nella parte superiore dello schermo. Se desiderate centrare l'immagine azionate il tasto X (↓) più volte fino a portare l'origine nella posizione più opportuna, quindi premete un tasto qualsiasi per la stampa (vedi figura 4). ■

Listato 1. *Programma per la stampa di grafici in alta risoluzione.*

```
100 REM   PERSONAL SOFTWARE
110 REM  *******************
120 REM  *                 *
130 REM  * GRAFICI AD ALTA *
140 REM  *                 *
150 REM  *   RISOLUZIONE   *
160 REM  *                 *
170 REM  *******************
180 REM  di: Sergio Borsani
190 REM  Tel.:(0436)3036
200 REM  versione: TI BASIC
210 CALL CLEAR
220 PRINT "  GRAFICO DI UNA FUNZIONE":::::::::::
230 DIM S$(14,14),P(240),ESA$(16),E$(15)
240 RESTORE 300
250 FOR J=0 TO 14
260 FOR K=0 TO 14
270 READ S$(J,K)
280 NEXT K
290 NEXT J
300 DATA 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E
310 DATA 1,1,3,3,5,5,7,7,9,9,B,B,D,D,F
320 DATA 2,3,2,3,6,7,6,7,A,B,A,B,E,F,E
330 DATA 3,3,3,3,7,7,7,7,B,B,B,B,F,F,F
340 DATA 4,5,6,7,4,5,6,7,C,D,E,F,C,D,E
350 DATA 5,5,7,7,5,5,7,7,D,D,F,F,D,D,F
360 DATA 6,7,6,7,6,7,6,7,E,F,E,F,E,F,E
370 DATA 7,7,7,7,7,7,7,7,F,F,F,F,F,F,F
380 DATA 8,9,A,B,C,D,E,F,8,9,A,B,C,D,E
390 DATA 9,9,B,B,D,D,F,F,9,9,B,B,D,D,F
400 DATA A,B,A,B,E,F,E,F,A,B,A,B,E,F,E
410 DATA B,B,B,B,F,F,F,F,B,B,B,B,F,F,F
420 DATA C,D,E,F,C,D,E,F,C,D,E,F,C,D,E
430 DATA D,D,F,F,D,D,F,F,D,D,F,F,D,D,F
440 DATA E,F,E,F,E,F,E,F,E,F,E,F,E,F,E
450 REM
460 REM     CARATTERI PER
470 REM     DISEGNARE GLI
480 REM     ASSI
490 REM
500 Z$(1)="0101010101010101"
510 Z$(2)="FF00000000000000"
520 Z$(3)="FF01010101010101"
530 CALL CHAR(33,Z$(1))
540 CALL CHAR(34,Z$(2))
550 CALL CHAR(35,Z$(3))
560 REM
570 REM        ISTRUZIONI
580 REM
590 CALL CLEAR
600 PRINT "  DEFINIRE LA FUNZIONE ALLA"
610 PRINT "LINEA 850 DEL PROGRAMMA":::
620 PRINT "  SE LO AVETE GIA' FATTO "
630 PRINT "SCRIVETE GLI ESTREMI"
640 PRINT "DELL'INTERVALLO DI"
650 PRINT "DEFINIZIONE (X1<X2).":: 
660 PRINT "  IL GRAFICO E' DIMENSIONATO"
670 PRINT "AUTOMATICAMENTE":::
680 INPUT "X1 = ":X1
690 INPUT "X2 = ":X2
700 IF X1<X2 THEN 740
710 PRINT
720 PRINT "DEVE ESSERE X1<X2, RIPROVA":::
730 GOTO 680
740 TRSX=0
750 TRSY=0
760 TRSX0=0
770 TRSY0=0
780 CALL CLEAR
790 PRINT TAB(7);"DIMENSIONAMENTO":::::
800 PRINT TAB(11);"RUNNING"::::::
810 REM
820 REM  ******************
830 REM
840 REM
850 DEF Y=X^3-3*X
860 REM
870 REM
880 REM        FUNZIONE
890 REM
900 REM  *****************
910 REM
920 UX=(X2-X1)/240
930 X=X1
940 MAX=Y
950 MIN=Y
960 FOR I=1 TO 240
970 X=X1+I*UX
980 IF Y<=MAX THEN 1000
990 MAX=Y
```

Grafici ad alta risoluzione con il TI99/4A

Seguito listato 1.

```
1000 IF Y>=MIN THEN 1020
1010 MIN=Y
1020 NEXT I
1030 UY=(MAX-MIN)/176
1040 IF UX>UY THEN 1070
1050 U=UY
1060 GOTO 1080
1070 U=UX
1080 IF U=0 THEN 2920
1090 X10=-X1/UX
1100 Y10=MAX/U
1110 COLO0=INT(X10/8)+1
1120 RIGA0=INT(Y10/8)+1
1130 X0=COLO0*8
1140 Y0=RIGA0*8+1
1150 INIZIO=0
1160 FOR I=1 TO 240
1170 X=(I-X0+7)*U
1180 IF (X>=X1)*(X<=X2)THEN 1210
1190 P(I)=0
1200 GOTO 1220
1210 P(I)=INT(Y0-Y/U)
1220 IF P(I)<1 THEN 1270
1230 IF P(I)>192 THEN 1270
1240 IF INIZIO<>0 THEN 1280
1250 INIZIO=I
1260 GOTO 1280
1270 IF INIZIO<>0 THEN 1290
1280 NEXT I
1290 FINE=I
1300 REM
1310 REM     ASSI CARTESIANI
1320 REM
1330 CALL CLEAR
1340 PRINT "     ESEGUITO IL GRAFICO,"
1350 PRINT "  PER USCIRE DAL PROGRAMMA"
1360 PRINT "PREMERE IL TASTO (Q), OPPURE"
1370 PRINT TAB(10);"FCTN(4).":: 
1380 PRINT " PRIMA DELLA STAMPA SI PUO'"
1390 PRINT "  CAMBIARE L'ORIGINE DEGLI"
1400 PRINT " ASSI CARTESIANI PREMENDO I"
1410 PRINT "     TASTI CON LE FRECCE.":: 
1420 PRINT "     PREMERE UN TASTO PER"
1430 PRINT TAB(10);"CONTINUARE.":::::
1440 CALL KEY(0,KEY,STATUS)
1450 IF STATUS=0 THEN 1440
1460 CALL CLEAR
1470 COLO0=COLO0+TRSX0
1480 RIGA0=RIGA0+TRSY0
1490 RIGA01=RIGA0+1
1500 IF (RIGA01<1)+(RIGA01>24)THEN 1540
1510 CALL HCHAR(RIGA01,2,34,30)
1520 FLAG=0
1530 GOTO 1550
1540 FLAG=1
1550 IF (COLO0<1)+(COLO0>32)THEN 1590
1560 CALL VCHAR(2,COLO0,33,22)
1570 FLAG=0
1580 GOTO 1600
1590 FLAG=1
1600 IF FLAG=1 THEN 1620
1610 CALL HCHAR(RIGA01,COLO0,35)
1620 GOSUB 2920
1630 IF TRASLA=1 THEN 1460
1640 DATA 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F
1650 RESTORE 1640
1660 FOR J=0 TO 15
1670 READ E$(J)
1680 NEXT J
1690 NUMCAR=35
1700 GOSUB 2850
1710 P1=P(INIZIO)
1720 RIGA1=INT((P1-1)/8)+1
1730 COLO1=INT((INIZIO-1)/8)+1
1740 NRC=P1-INT((P1-1)/8)*8
1750 NCC=INIZIO-INT((INIZIO-1)/8)*8
1760 NB1=NRC*2
1770 IF NCC>4 THEN 1800
1780 NB1=NB1-1
1790 GOTO 1810
1800 NCC=NCC-4
1810 ESADEC1=2^(4-NCC)
1820 FOR PIX=INIZIO+1 TO FINE-1
1830 P2=P(PIX)
1840 RIGA2=INT((P2-1)/8)+1
1850 COLO2=INT((PIX-1)/8)+1
1860 NRC=P2-INT((P2-1)/8)*8
1870 NCC=PIX-INT((PIX-1)/8)*8
1880 NB2=NRC*2
1890 IF NCC>4 THEN 1920
1900 NB2=NB2-1
1910 GOTO 1930
1920 NCC=NCC-4
1930 ESADEC2=2^(4-NCC)
1940 IF P1=P2 THEN 2150
1950 ESA$(NB1)=E$(ESADEC1)
1960 DIFF=P2-P1
1970 W=SGN(DIFF)
1980 VOLTE=ABS(DIFF)-1
1990 IF VOLTE=0 THEN 2100
2000 FOR Z=1 TO VOLTE
2010 NB1=NB1+W*2
2020 IF (NB1<1)+(NB1>16)THEN 2040
2030 GOTO 2080
```

Seguito listato 1.

```
2040 GOSUB 2400
2050 GOSUB 2850
2060 RIGA1=RIGA1+W
2070 NB1=NB1-W*16
2080 ESA$(NB1)=E$(ESADEC1)
2090 NEXT Z
2100 IF RIGA1<>RIGA2 THEN 2120
2110 IF COLO1=COLO2 THEN 2260
2120 GOSUB 2400
2130 GOSUB 2850
2140 GOTO 2260
2150 REM  *** STESSA PIX-RIGA ***
2160 IF COLO1=COLO2 THEN 2210
2170 ESA$(NB1)=E$(ESADEC1)
2180 GOSUB 2400
2190 GOSUB 2850
2200 GOTO 2260
2210 IF NB1=NB2 THEN 2240
2220 ESA$(NB1)=E$(ESADEC1)
2230 GOTO 2260
2240 ESADEC1=ESADEC1+ESADEC2
2250 GOTO 2270
2260 ESADEC1=ESADEC2
2270 P1=P2
2280 RIGA1=RIGA2
2290 COLO1=COLO2
2300 NB1=NB2
2310 NEXT PIX
2320 CALL KEY(0,KEY,STATUS)
2330 IF STATUS=0 THEN 2320
2340 IF KEY=81 THEN 3200
2350 IF KEY=113 THEN 3200
2360 GOTO 2320
2370 REM
2380 REM       STAMPA
2390 REM
2400 CARAT$=ESA$(1)
2410 FOR I=2 TO 16
2420 CARAT$=CARAT$&ESA$(I)
2430 NEXT I
2440 NUMCAR=NUMCAR+1
2450 IF NUMCAR>159 THEN 2320
2460 RIGA=RIGA1+TRSY
2470 IF (RIGA<1)+(RIGA>24)THEN 2550
2480 COLONNA=COLO1+1+TRSX
2490 IF (COLONNA<1)+(COLONNA>32)THEN 2550
2500 CALL GCHAR(RIGA,COLONNA,CC)
2510 IF CC=0 THEN 2530
2520 GOSUB 2590
2530 CALL CHAR(NUMCAR,CARAT$)
2540 CALL HCHAR(RIGA,COLONNA,NUMCAR)
2550 RETURN
2560 REM
2570 REM   SOMMA CARATTERI
2580 REM
2590 IF CC<33 THEN 2800
2600 IF CC>35 THEN 2800
2610 C$=" "
2620 FOR I=1 TO 16
2630 L1=ASC(SEG$(CARAT$,I,1))
2640 L2=ASC(SEG$(Z$(CC-32),I,1))
2650 IF L1>60 THEN 2680
2660 L1=L1-48
2670 GOTO 2690
2680 L1=L1-55
2690 IF L2>60 THEN 2720
2700 L2=L2-48
2710 GOTO 2730
2720 L2=L2-55
2730 IF L1=15 THEN 2770
2740 IF L2=15 THEN 2770
2750 C$=C$&S$(L1,L2)
2760 GOTO 2780
2770 C$=C$&"F"
2780 NEXT I
2790 CARAT$=SEG$(C$,2,16)
2800 RETURN
2810 REM
2820 REM      AZZERAMENTO
2830 REM          ESA$
2840 REM
2850 FOR I=1 TO 16
2860 ESA$(I)="0"
2870 NEXT I
2880 RETURN
2890 REM
2900 REM     TRASLAZIONE
2910 REM
2920 CALL KEY(0,TASTO,ST)
2930 IF ST=0 THEN 2920
2940 IF TASTO=69 THEN 3000
2950 IF TASTO=88 THEN 3040
2960 IF TASTO=83 THEN 3080
2970 IF TASTO=68 THEN 3120
2980 TRASLA=0
2990 GOTO 3160
3000 TRSY=TRSY-1
3010 TRSY0=-1
3020 TRSX0=0
3030 GOTO 3150
3040 TRSY=TRSY+1
3050 TRSY0=1
3060 TRSX0=0
3070 GOTO 3150
3080 TRSX=TRSX-1
3090 TRSY0=0
3100 TRSX0=-1
3110 GOTO 3150
3120 TRSX=TRSX+1
3130 TRSY0=0
3140 TRSX0=1
3150 TRASLA=1
3160 RETURN
3170 REM
3180 REM      FINE
3190 REM
3200 CALL CLEAR
3210 END
```

PERSONAL SOFTWARE

Othello per ZX Spectrum

Una nuova versione per Sinclair del popolarissimo gioco di strategia

di *Stefano Cerutti*

Il programma del listato 1 è una versione per lo ZX Spectrum dell'ormai famoso gioco Othello.

La versione presentata occupa 9585 byte e permette di simulare una vera partita tra due giocatori; anche la presentazione grafica è stata molto curata.

Dopo il RUN, appare su uno sfondo giallo una pioggia di "Ferma il registratore".

Iniziando il gioco con la pressione di un tasto, viene stampato il titolo e vengono chiesti tramite INPUT i nomi dei due giocatori, il bianco e il nero, che possono essere al massimo di 13 caratteri.

Per chi non abbia mai giocato il programma può stampare le istruzioni usando una particolare routine di stampa che simula il funzionamento di una telescrivente. Interessante anche come vengono trattati i "data".

Dopo essersi accertato che le istruzioni siano state comprese il programma prepara la zona di gioco, un quadrato verde di otto caselle per lato, con le ascisse numerate da uno a otto, e le ordinate contrassegnate dalle lettere da A a H.

La prima mossa è sempre del giocatore bianco, che deve inserire le coordinate del quadratino dove intende piazzare il suo gettone: bisogna indicare sempre prima la lettera, poi il numero (ad esempio b5, a8).

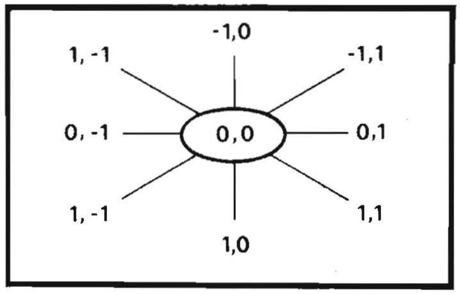


Figura 1. *Il programma esamina tutte le direzioni intorno alla casella dove viene posizionato il gettone.*

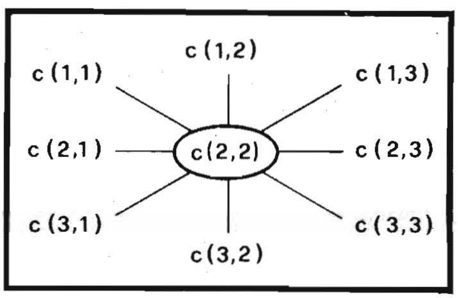


Figura 2. *In questo esempio vengono esplorate le direzioni intorno alla (2,2).*

Dopo circa due secondi i gettoni che devono essere girati cambiano colore con un ticchettìo caratteristico, e il calcolatore è pronto per ricevere la mossa dell'altro giocatore.

La situazione di gioco (il numero di gettoni del giocatore bianco e il numero di quelli del giocatore nero) e l'ultima mossa eseguita sono sempre visualizzati e aggiornati.

Le mosse non consentite vengono segnalate dalla scritta "MOSSA NON CORRETTA", da un beep e chieste di nuovo.

Non è consentito piazzare un gettone in una posizione già occupata o fare mosse che non facciano rovesciare almeno un gettone avversario.

Le mosse si alternano così fino a quando tutta la zona di gioco non è completamente piena; ed allora viene proclamato vincitore colui che avrà più gettoni del proprio colore in campo.

Infine viene chiesto se si intende continuare a giocare se si desidera abbandonare il programma.

Il carattere grafico che rappresenta il gettone corrisponde alla lettera "A".

Fate quindi attenzione ad inserire alle linee 40, 344, 355 il carattere giusto, cioè la "A" in modo "graphics".

Da notare il largo uso della variabile di sistema SCR CT che occupa il byte 23692 e della funzione TO.

Detta variabile di sistema contiene il numero aumentato di uno degli scrolling che il sistema esegue prima di chiedere "scroll?".

Se mantenuta, per esempio, a 255, ciò evita al sistema di stampare "scroll?", fermare il programma e aspettare una risposta.

Con questo semplice accorgimento è stato ottenuto lo scrolling automatico senza interruzione.

La funzione TO per l'estrazione o la definizione di sottostringhe è caratteristica del BASIC Sinclair.

Questa permette di ottenere gli stessi effetti delle funzioni-stringa dei BASIC standard, LEFT$, RIGHT$, MID$ e TL$.

Per usare questa preziosa funzione è sufficiente specificare la stringa sulla quale si intende operare e, tra parentesi, il carattere di inizio e di fine della sottostringa separate da TO.

Per esempio, se a$ = "123456789", a$ (5 TO 8) dà come risultato "5678".

Con un BASIC standard, per ottenere lo stesso risultato si sarebbe dovuto scrivere: MID$ (a$, 5, 4).

Nel digitare il listato, i più pigri possono astenersi dal ricopiare le linee da 560 a 620 comprese, tutti i

Othello per ZX Spectrum

DATA da riga 741 fino alla fine del listato e tutti i REMarks.

Il programma sarà così privato dalle istruzioni, ma occuperà meno memoria (circa 5734 byte) e girerà con meno grafica!

Sperando che il programma vi piaccia, auguro a tutti buon divertimento! ■

Listato 1. *Il programma Othello.*

```
   1 REM ************************
   2 REM *                      *
   3 REM *    O T E L L O       *
   4 REM *                      *
   5 REM * by Stefano Cerutti   *
   6 REM *                      *
   7 REM ************************
   8 BORDER 6
   9 PAPER 6
  10 INK 1
  11 CLS
  12 GO TO 15
  13 GO SUB 485
  14 GO TO 20
  15 GO SUB 481
  16 REM ************************
  17 REM *stampa campo da gioco*
  18 REM ************************
  20 CLS
  21 FOR t=1 TO 6
  25 PRINT AT 1,t+1;t;AT t+1,1;C
HR$ (t+96); INK 4;"[redacted]"
  30 NEXT t
  35 GO SUB 459
  40 PRINT AT 5,5; INK 7; PAPER
4;"A"; INK 0;"A";AT 6,5;"A"; INK
 7;"A"
  45 LET bianchi=2
  50 LET neri=2
  55 LET f=0
  60 LET tot=4
  65 DIM c(3,3)
  75 DIM x(20)
  80 DIM y(20)
  85 LET k$=""
  86 LET j$=""
  87 IF bianchi<10 THEN LET k$="
 "
  88 IF neri<10 THEN LET j$=" "
  89 PRINT AT 14,2;"BIANCHI= ";k
$;bianchi;AT 14,15;"NERI= ";j$;n
eri
 100 IF f=0 THEN LET f=7: PRINT
AT 18,2; INVERSE 1;"MUOVE ";b$;"
 ,IL BIANCO": GO TO 115
 105 LET f=0
 110 PRINT AT 18,2; INVERSE 1;"M
UOVE ";n$;" ,IL NERO  "
 115 INPUT m$
 120 LET cz=0
 125 IF m$="xx" THEN GO TO 100
 130 IF m$="end" THEN GO TO 390
 135 IF LEN m$<>2 THEN GO TO 115
 136 IF CODE m$(2)>56 OR CODE m$
(2)<49 OR CODE m$<97 OR CODE m$>
104 THEN GO TO 115
 140 PRINT AT 9,11;m$
 145 PRINT AT 4,13; FLASH 1;"ATT
ENDERE,  PREGO."
 150 LET cx=VAL m$(2)+1
 155 LET cy=(CODE m$)-95
 160 LET c2=0
 165 IF ATTR (cy,cx)<>52 THEN PR
INT AT 4,13; INVERSE 1;"MOSSA NO
N CORRETTA": BEEP 2,0: PRINT AT
4,13;t$( TO 18): GO TO 115
 166 REM ************************
 167 REM * carica in c(y,x) le *
 168 REM * direzioni di gioco  *
 169 REM ************************
 170 FOR y=1 TO 3
 175 FOR x=1 TO 3
 180 IF x=2 AND y=2 THEN NEXT x
 185 LET a=ATTR (cy+y-2,cx+x-2)
 190 IF a<>39-f THEN LET c(y,x)=
0: GO TO 205
 195 LET c(y,x)=1
 200 LET c2=c2+1
 205 NEXT x
 210 NEXT y
 215 IF c2=0 THEN PRINT AT 4,13;
 INVERSE 1;"MOSSA NON CORRETTA":
 BEEP 2,0: PRINT AT 4,13;t$( TO
18): GO TO 115
 219 LET b1=bianchi
 220 LET n1=neri
 221 REM ************************
 222 REM *legge c(y,x),elabora,*
 223 REM *  analizza,punteggi, *
 224 REM *   deposita in x(cz),*
 225 REM * y(cz) le coordinate *
 226 REM *dei gettoni da girare*
 227 REM ************************
 229 FOR y=1 TO 3
 230 FOR x=1 TO 3
 235 IF c(y,x)=0 THEN NEXT x: NE
XT y: GO TO 330
 240 LET ax=x-2
 245 LET ay=y-2
 250 LET cx1=cx
 255 LET cy1=cy
 265 LET rp=cz
 270 LET cz=cz+1
```

PRINCIPALI ROUTINE UTILIZZATE

20-40	Preparazione del video, con chiamata al sottoprogramma di linea 459 che definisce il carattere del gettone.
45-86	Inizializzazione variabili e DIMensionamento vettori.
89-145	Stampa i punteggi, sceglie chi deve muovere, richiede la mossa, controlla la sua validità ed esattezza.
160-220	Analizza la posizione del gettone appena depositato, sceglie le direzioni su cui lavorare e deposita i risultati di questa analisi in c(3,3).
229-315	Legge il vettore c(3,3), elabora, deposita le coordinate dei gettoni da girare in x(20) e y(20).
330-380	Aggiorna i punteggi, stampa i gettoni, aggiorna il numero totale dei gettoni e controlla se la partita è finita.
390-455	Routine cui si accede a partita conclusa: proclamazione del vincitore e richiesta di una nuova partita.
459-480	Routine di definizione della matrice del gettone.
481-625	Presentazione, chiede il nome dei due giocatori, richiede e stampa le istruzioni.
635-735	DATA per la stampa della presentazione (titolo).
740	DATA per la definizione del gettone.
745-1045	DATA per le istruzioni.

VARIABILI USATE

t, l, y, x	Variabili di controllo nei cicli FOR...TO...NEXT.
bianchi	Numero dei gettoni bianchi.
neri	Numero dei gettoni neri.
f	Se f = 7, sta giocando il bianco, se f = 0 sta giocando il nero.
tot	Numero totale di gettoni presenti sul campo da gioco.
cx, cy	Coordinate del gettone appena giocato.
bits	Contiene uno degli otto byte che occorrono per la definizione del carattere.
x1, y1, x2, y2	Variabili usate per stampare il titolo mediante 44 serie di PLOT e DRAW.
ax, ay, cx1, cy1, rp	Variabili usate durante l'analisi delle direzioni di gioco.
cz	Contiene il numero di gettoni da capovolgere in una mossa, cioè quelli che vengono sottratti all'avversario.
c2	Contiene un numero compreso tra 0 e 8, che è quello delle direzioni dove è possibile girare dei gettoni. Se contiene zero la mossa non è valida.
a	Contiene gli attributi di una delle otto posizioni adiacenti al gettone appena giocato.
a1	Simile ad a, lavora insieme ad ax, ay.
b1	Prima di ogni mossa, viene posto uguale ai bianchi.
n1	Come b1, se dopo una mossa tutti e due sono rimasti uguali ai loro corrispettivi, si ha il messaggio di errore perchè significa che la mossa non ha fatto girare nessun gettone.

VARIABILI STRINGA

t$	Contiene sempre 30 spazi e serve un po' ovunque.
m$	Coordinate del gettone appena giocato.
d$	Se contiene "no" si desidera abbandonare il programma: questo provoca la cancellazione automatica di tutta la memoria, lasciandola pulita.
o$	Se contiene una qualsiasi stringa diversa da "n", si desiderano le istruzioni.
f$	Contiene una linea di istruzioni da stampare.
r$	Se contiene "n" si desidera rivedere le istruzioni.
k$	Se il numero dei gettoni bianchi ha una sola cifra, contiene uno spazio, altrimenti è una stringa nulla.
j$	Lo stesso di k$, ma per il nero.

Othello per ZX Spectrum

Seguito listato 1.

```
275 LET a1=ATTR (cy1+ay,cx1+ax)
280 IF a1<>39 AND a1<>32 THEN LET cz=rp: NEXT x: NEXT y: GO TO 330
285 IF a1=32+f THEN LET cz=cz-1: NEXT x: NEXT y: GO TO 330
295 LET x(cz)=cx1+ax
300 LET y(cz)=cy1+ay
305 LET cx1=cx1+ax
310 LET cy1=cy1+ay
315 GO TO 270
316 REM ************************
317 REM * aggiornamento punti  *
318 REM ************************
330 IF cz=0 THEN PRINT AT 4,13; INVERSE 1;"MOSSA NON CORRETTA": BEEP 2,0: PRINT AT 4,13;t$( TO 18): GO TO 115
333 IF f=0 THEN LET neri=neri+1+cz: LET bianchi=bianchi-cz: GO TO 338
335 LET neri=neri-cz
336 LET bianchi=bianchi+cz+1
338 REM ************************
339 REM *  stampa gettone e    *
340 REM *  cambio di colore    *
341 REM ************************
344 PRINT AT 4,13;t$( TO 18);AT cy,cx; PAPER 4; INK f;"A"
345 BEEP .01,20
350 FOR t=1 TO cz
355 PRINT AT y(t),x(t); PAPER 4; INK f;"A"
360 BEEP .01,20
365 NEXT t
367 IF bianchi=0 OR neri=0 THEN GO TO 390
370 LET tot=tot+1
375 IF tot=64 THEN GO TO 390
380 GO TO 85
381 REM ************************
382 REM *    proclamazione     *
383 REM * vincitore o parita'  *
384 REM ************************
390 PRINT AT 14,2;t$;AT 18,2;t$;AT 4,13;t$( TO 18)
395 FLASH 1
400 BRIGHT 1
405 IF bianchi>neri THEN LET v$=b$: LET p$=n$: GO TO 420
410 IF bianchi=neri THEN GO TO 430
415 LET p$=b$
417 LET v$=n$
420 PRINT AT 12,3; INK 2;"BRAVO, ";v$;"!          SEI RIUSCITO A BATTERE ";p$;" PER ";bianchi;" A ";neri;" !"
425 GO TO 435
430 PRINT AT 12,3; INK 2;"BRAVI ";u$;" E ";v$;"! AVETE PAREGGIATO!"
431 REM ************************
432 REM *  richiesta di una    *
433 REM *    nuova partita     *
434 REM ************************
435 INPUT "UN'ALTRA PARTITA?";d$
440 IF d$="no" THEN NEW
445 FLASH 0
450 BRIGHT 0
455 RUN 15
456 REM ************************
457 REM *  def. carattere      *
458 REM ************************
459 RESTORE 740
460 FOR t=0 TO 7
465 READ bits
470 POKE USR "a"+t,bits
475 NEXT t
480 RETURN
481 IF INKEY$<>"" THEN GO TO 485
482 PRINT TAB RND*19; FLASH 1;"STOP the tape"
483 POKE 23692,255
484 GO TO 481
485 CLS
486 REM ************************
487 REM * stampa presentazione*
488 REM ************************
490 FOR t=1 TO 44
495 READ x1
500 READ y1
505 PLOT INK 3;x1*8-1,y1*8-1
510 READ x2
515 READ Y2
520 DRAW INK 3;x2*8,y2*8
525 NEXT t
530 PRINT AT 14,7; INK 4; FLASH 1; BRIGHT 1;"by Stefano Cerutti"
535 PAUSE 0
536 DIM t$(30)
537 DIM b$(13)
538 DIM n$(13)
540 INPUT "NOME GIOCATORE BIANCO          (MAX.13 CHAR.)";u$
545 IF LEN u$>13 THEN GO TO 540
550 INPUT "NOME GIOCATORE NERO            (MAX.13 CHAR.)";v$
555 IF LEN v$>13 THEN GO TO 550
556 LET b$(INT ((13-(LEN u$))/2) TO INT ((13-(LEN u$))/2)+LEN u$)=u$
557 LET n$(INT ((13-(LEN v$))/2) TO INT ((13-(LEN v$))/2)+LEN v$)=v$
560 INPUT "VOLETE LE ISTRUZIONI ?(s/n)";o$
565 IF o$="n" THEN RETURN
570 POKE 23692,255
575 CLS
576 REM ************************
577 REM *  stampa istruzioni   *
578 REM ************************
579 RESTORE 745
580 FOR t=1 TO 60
585 READ f$
```

VETTORI USATI

x(20) Ascissa dei gettoni che devono cambiare colore conseguentemente ad una mossa. È formato da venti variabili perchè questo è, teoricamente, il numero massimo di gettoni che è possibile girare in una sola mossa.
È molto raro ma possibile che si verifichi questa situazione.

y(20) Ordinata dei gettoni da girare, usata insieme a x(20).

c(3,3) Questo vettore a due dimensioni costituisce praticamente il "cuore" del programma.
Esso permette di sapere subito le direzioni di gioco: se la variabile corrispondente è settata (contiene uno) vuol dire che rappresenta una direzione da analizzare, altrimenti contiene zero e la direzione corrispettiva viene ignorata.
Dalla figura 2 appare chiaro che con due cicli FOR...TO...NEXT concatenati è possibile controllare tutte le otto direzioni.
Naturalmente, dato che un vettore non può avere indice − 1 o 0, è stato tutto aumentato di due (figura 2).
La posizione 2,2 non viene esaminata, non essendo una direzione ma la posizione del gettone appena giocato: da qui si capisce il perchè delle linee 180, 185, 240 e 245.

Seguito listato 1.

```
587 PRINT
588 PRINT
589 POKE 23692,255
590 FOR l=1 TO LEN f$
591 IF CODE f$(l)=32 THEN GO TO 600
595 BEEP .01,1
600 PRINT f$(l);
605 NEXT l
607 BEEP .08,25
610 NEXT t
612 PAUSE 3000
615 INPUT "E'TUTTO CHIARO ?(s/n)";r$
620 IF r$="n" THEN GO TO 570
623 CLS
625 RETURN
626 REM ************************
627 REM * DATA per titolo      *
628 REM ************************
635 DATA 7,20,0,-5,7,15,3,0,10,15
640 DATA 0,5,10,20,-3,0,8,19,0,-3
645 DATA 6,16,1,0,9,16,0,3,9,19,-1
650 DATA 0,13,19,-1,0,12,19,0,-1
655 DATA 12,15,-1,0,11,15,0,4,11
660 DATA 19,-1,0,10,20,3,0,13,23
665 DATA 0,-5,13,15,3,0,16,15,0,1
670 DATA 16,16,-2,0,14,16,0,1,11
675 DATA 17,1,0,15,17,0,1,15,16,-1
680 DATA 0,14,18,0,1,14,19,2,0,16
685 DATA 19,0,1,13,20,4,0,17,20,0
690 DATA -4,17,16,2,0,19,16,0,-1
695 DATA 19,15,-3,0,16,16,0,3,19
700 DATA 20,0,-5,19,15,3,0,19,23
705 DATA 1,0,20,20,0,-4,20,16,2,0
710 DATA 22,15,0,5,22,20,3,0,25,20
715 DATA 0,-5,25,15,-3,0,23,19,1,0
720 DATA 24,19,0,-3,24,16,-1,0,33
725 DATA 16,0,3
736 REM ************************
737 REM * DATA def. carattere *
738 REM ************************
740 DATA 0,60,126,126,126,126,60,0
741 REM ************************
742 REM * DATA per istruzioni *
743 REM ************************
745 DATA t$( TO 13)+"Otello"+t$( TO 13)
750 DATA t$( TO 13)+"______"+t$( TO 13)
755 DATA t$
760 DATA t$( TO 7)+"by Stefano Cerutti"+t$( TO 7)
765 DATA t$,t$
770 DATA "Questa e' una versione del      "
776 DATA "popolare gioco per due persone  "
780 DATA "chiamato Othello.               "
785 DATA "Suo scopo e' quello di imprigio-"
790 DATA "nare i gettoni avversari,       "
795 DATA "trasformarli in gettoni del     "
800 DATA "proprio colore e concludere la  "
805 DATA "partita con un numero di gettoni"
```

Seguito listato 1.

```
810 DATA "maggiore di quello dell' avver-"
815 DATA "sario.                          "
820 DATA "Dopo aver inserito i vostri nomi"
825 DATA "il giocatore bianco dovra' inse-"
830 DATA "rire le coordinate della posi-"
835 DATA "zione dove intende piazzare il  "
840 DATA "suo gettone, prima la lettera , "
845 DATA "poi il numero.                  "
850 DATA "In seguito le mosse si alterne-"
855 DATA "ranno,bianco e nero.            "
860 DATA "Il numero di gettoni, il turno  "
865 DATA "di mossa e la mossa stessa ver-"
870 DATA "ranno sempre visualizzati.      "
875 DATA "Il ribaltamento dei gettoni e'  "
880 DATA "automatico.                     "
885 DATA "Se un giocatore con una sua mos-"
890 DATA "sa non riuscira' a girare       "
900 DATA "neanche un gettone avversario   "
905 DATA "verra' visualizzato il messaggio"
910 DATA "di errore.                      "
915 DATA "In tal caso dovra' inserire le  "
920 DATA "   coordinate di un'altra mossa"
925 DATA "se possibile, oppure digitare   "
930 DATA " 'xx' per saltare il turno.     "
935 DATA "La partita ha termine quando:   "
940 DATA "SUL CAMPO DI GIOCO SONO PRESENTI"
945 DATA "64 GETTONI,                     "
950 DATA "la scacchiera e' piena e automa-"
955 DATA "ticamente verra' proclamato vin-"
960 DATA "citore il giocatore con piu'get-"
965 DATA "toni del proprio colore.        "
970 DATA "NESSUNO DEI GIOCATORI PUO' FARE "
975 DATA "NUOVE MOSSE.                    "
980 DATA "sul campo di gioco sono presenti"
985 DATA "solo gettoni di un colore.      "
990 DATA "In tal caso,invece di una mossa,"
995 DATA "occore digitare 'end' e         "
1000 DATA "seguira' la proclamazione del   "
1005 DATA "vincitore.                      "
1010 DATA t$,t$
1015 DATA "E' comunque possibile terminare "
1020 DATA "la partita in ogni momento ,    "
1025 DATA "digitanto 'end' al posto di una "
1030 DATA "mossa qualsiasi.                "
1040 DATA t$( TO 7)+"Buon Divertimento !!!"+t$( TO 7)
1045 REM *****THE**END**********
```

Dama cinese e Motocross

Due giochi per ZX81

di *Claudio Driussi*

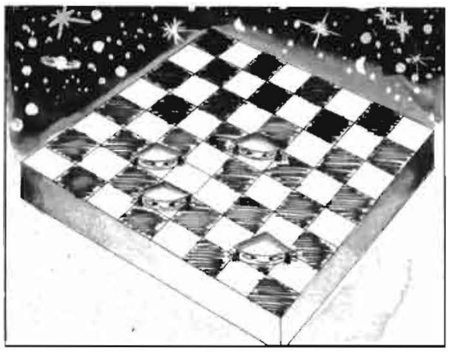
Dama cinese

Questa Dama cinese è la versione per ZX81 del famoso solitario di origine orientale, le regole sono semplici: all'inizio ci sono, su una scacchiera a forma di croce, 32 pedine che vanno eliminate "mangiandole" allo stesso modo della dama tradizionale, ma a differenza di questa non è consentito muovere una pedina senza mangiarne un'altra. Scopo del gioco è quello di mangiare tutte le pedine ad eccezione di una che deve rimanere nel centro della scacchiera.

Per quanto riguarda la grafica si scrivono direttamente nella memoria video i codici delle pedine. Per giocare prima si inseriscono le coordinate della pedina da muovere e poi la direzione sulla base dei tasti che appaiono sullo schermo; per abbandonare si inserisce 0 alla richiesta di mossa.

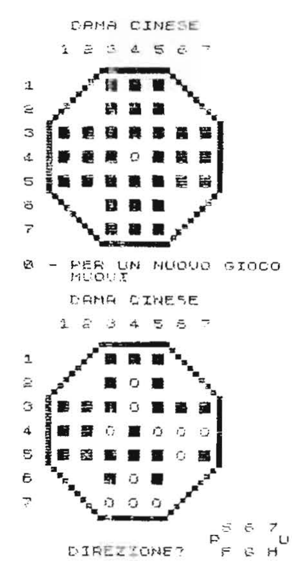
Motocross

Ecco un gioco di animazione, con soggetto il Motocross. L'idea di questo programma è nata da un programma per VIC 20 del numero 2 di P.S.; data la grande differenza tra ZX81 e VIC 20 i programmi sono completamente diversi. Il programma simula la corsa di un motociclista che deve saltare una serie di ostacoli disseminati lungo il cammino; per saltare si preme il tasto "7". Alla fine della corsa il calcolatore legge, in FAST, la memoria di schermo per sapere quanti ostacoli sono rimasti (codice 4) ed in base al risultato ed alla precisione dei salti calcola il punteggio. Il programma tiene inoltre conto del massimo punteggio raggiunto da quando è stato acceso il calcolatore; non si possono comunque fare più di 30 punti. ■



Listato 1. *Il programma del gioco Dama cinese ed un esempio di gioco.*

```
  10 PRINT TAB 10;"DAMA CINESE
  30 PRINT
  40 PRINT TAB 9;"1 2 3 4 5 6 7
  50 PRINT
  60 PRINT TAB 5;"
  70 PRINT TAB 5;"1     ■ ■ ■
  80 PRINT TAB 6;"
  90 PRINT TAB 6;"2     ■ ■ ■
 100 PRINT TAB 5;"
 110 PRINT TAB 6;"3 ■ ■ ■ ■ ■ ■ ■
 120 PRINT TAB 6;"
 130 PRINT TAB 6;"4 ■ ■ ■ O ■ ■ ■
 140 PRINT TAB 6;"
 150 PRINT TAB 6;"5 ■ ■ ■ ■ ■ ■ ■
 160 PRINT TAB 6;"
 170 PRINT TAB 6;"6     ■ ■ ■
 180 PRINT TAB 6;"
 190 PRINT TAB 6;"7     ■ ■ ■
 200 PRINT TAB 6;"
 210 LET K=175+PEEK 16396+PEEK 1
6397*256
 215 LET N=32
 217 PRINT AT 20,6;"0 - PER UN N
UOVO GIOCO"
 220 PRINT AT 21,10;"MUOVI"
 230 INPUT M
 235 IF M=0 THEN GOTO 800
 240 LET Y=INT (M/10)
 250 LET X=M-Y*10
 260 IF Y<1 OR Y>7 OR X<1 OR X>7
 THEN GOTO 230
 270 PRINT AT 20,6;"
     ";AT 21,10;"       "
 280 LET P=K+56*(Y-1)+2*(X-1)
 290 IF PEEK P=128 THEN GOTO 320
 300 PRINT AT 20,6;"COORDINATE E
RRATE      "
 310 GOTO 220
 320 PRINT AT 19,23;"5 6 7"
 330 PRINT AT 20,22;"R     U"
 340 PRINT AT 21,23;"F G H"
 350 PRINT AT 21,10;"DIREZIONE?"
 360 POKE P,189
 370 LET S=0
 380 INPUT D$
 390 PRINT AT 19,23;"     "
 400 PRINT AT 20,22;"       "
 410 PRINT AT 21,10;"
     "
 420 IF D$<>"5" THEN GOTO 450
 430 LET S=-136
 440 LET O=-68
 450 IF D$<>"6" THEN GOTO 480
 460 LET S=-132
 470 LET O=-66
 480 IF D$<>"7" THEN GOTO 510
 490 LET S=-128
 500 LET O=-64
 510 IF D$<>"R" THEN GOTO 540
 520 LET S=-4
 530 LET O=-2
 540 IF D$<>"U" THEN GOTO 570
 550 LET S=4
 560 LET O=2
 570 IF D$<>"F" THEN GOTO 600
 580 LET S=128
 590 LET O=64
 600 IF D$<>"G" THEN GOTO 630
 610 LET S=132
 620 LET O=66
 630 IF D$<>"H" THEN GOTO 660
 640 LET S=136
 650 LET O=68
 660 IF S=0 THEN GOTO 320
 665 POKE P,128
 670 IF PEEK (P+O)=128 AND PEEK
(P+S)=52 THEN GOTO 700
 680 PRINT AT 20,6;"MOSSA NON CO
NSENTITA"
 690 GOTO 220
 700 POKE P,52
 710 POKE P+S,128
 720 POKE P+O,52
 730 LET N=N-1
 740 IF N=1 THEN GOTO 760
 750 GOTO 220
 760 PRINT AT 21,10;"HAI VINTO"
 770 IF INKEY$="" THEN GOTO 770
 780 CLS
 790 GOTO 1
 800 PRINT AT 21,3;"TI SONO RIMA
STE ";N;" PEDINE"
 810 IF INKEY$="" THEN GOTO 810
 820 CLS
 830 GOTO 1
9980 STOP
9990 SAVE "DAMA CINESE"
9999 GOTO 1
```



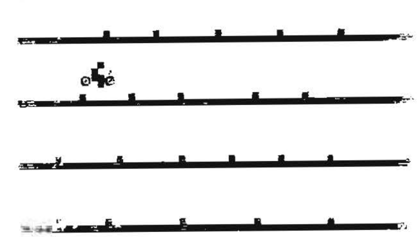
PERSONAL SOFTWARE

Dama cinese e Motocross



```
   5 LET PM=0
  10 PRINT TAB 9;"MOTOCROSS"
  20 LET A$="  "
  30 LET B$="  "
 100 FOR I=6 TO 21 STEP 5
 110 PRINT AT I,0;"
"
 120 NEXT I
 130 LET A=5
 135 LET B=2
 140 FOR O=1 TO 50
 150 LET B=B+INT (RND*3+4)
 160 IF B<26 THEN GOTO 210
 190 LET B=B-26
 200 LET A=A+5
 210 PRINT AT A,B;"■"
 220 IF A=20 AND B>21 THEN GOTO
240
 230 NEXT O
 240 LET P=51+O*3
 242 PRINT AT 4,0;A$
 243 PRINT AT 5,0;B$
 250 PRINT AT 2,0;"PRONTI"
 260 FOR I=1 TO 100
 270 NEXT I
 280 PRINT AT 2,0;"VIA     "
 285 LET P=P-.1
 290 IF INKEY$="" THEN GOTO 285
 320 LET F=0
 350 LET A=5
 360 LET B=0
 370 FOR J=1 TO 116
 380 LET B=B+1
 390 IF B<30 THEN GOTO 420
 392 PRINT AT A-2,29;"  "
 394 PRINT AT A-1,29;"   "
 395 PRINT AT A,29;"   "
 400 LET B=B-30
 410 LET A=A+5
 420 IF INKEY$="7" THEN GOTO 470
 425 PRINT AT A-1,B;A$
 430 PRINT AT A,B;B$
 440 IF F<=0 THEN PRINT AT A,B-1
;" "
 442 IF F=0 THEN GOTO 450
 444 PRINT AT A-2,B;"  "
 445 PRINT AT A-1,B-1;" "
 448 LET F=0
 450 NEXT J
 460 GOTO 610
 470 IF F=5 OR F=-1 THEN GOTO 57
0
 472 LET P=P-1
 475 PRINT AT A-2,B;A$
 480 PRINT AT A-1,B;B$
 490 PRINT AT A-1,B-1;" "
 500 IF F=0 THEN PRINT AT A,B-1;
"  "
 510 LET F=F+1
 550 NEXT J
 560 GOTO 610
 570 LET P=P-2
 575 PRINT AT A-1,B;A$
 580 PRINT AT A,B;B$
 590 PRINT AT A,B-1;" "
 591 IF F<>5 THEN GOTO 600
 592 PRINT AT A-2,B;"  "
 594 PRINT AT A-1,B-1;" "
 596 LET F=-1
 600 NEXT J
 610 PRINT AT 18,28;"    "
 620 PRINT AT 19,28;" ";A$
 630 PRINT AT 20,28;" ";B$
 640 LET K=PEEK 16396+PEEK 16397
*255
 645 LET OS=0
 646 FAST
 650 FOR I=0 TO 726
 660 IF PEEK (K+I)=4 THEN LET OS
=OS+1
 670 NEXT I
 680 LET P=INT P-(O-OS)*7
 685 IF P<0 THEN LET P=0
 688 IF PM<P THEN LET PM=P
 690 SLOW
 700 PRINT AT 2,0;"PUNTI ";P;" P
UNTEGGIO MASSIMO ";PM
 710 IF INKEY$="" THEN GOTO 710
 720 CLS
 730 GOTO 10
9989 STOP
9990 SAVE "MOTOCROS■"
9999 GOTO 1
```

```
VIA
```



Listato 2. *Il programma del gioco Motocross ed un esempio di gioco.*


in edicola
• Prezzi
• Caratteristiche
• Descrizioni
• Prove
ANNUARIO 84
DI VIDEO GIOCHI
L. 5.000
TUTTI I GIOCHI PER COMPUTER TUTTI I VIDEOGAMES
Descrizioni, caratteristiche, illustrazioni a colori e prezzi di oltre 500 giochi, accessori, computer.
• Tutti i videogames
• Tutte le consolle
• Tutti i giochini tascabili
• Tutti gli accessori
• Tutti i giocomputer
• Tutti a colori
Tutto... tutto...
ma proprio tutto.

Piani di ammortamento mutui con lo ZX81

PERSONAL SOFTWARE

Una lezione di economia per fare bene i conti con le rate

di *Angelo Motta*

Fra i vari tipi di finanziamenti che le banche concedono ai privati figurano i mutui ipotecari ed i prestiti personali.

I primi, assistiti da garanzia ipotecaria come indica il nome stesso, vengono rilasciati per costruzione, acquisto o ristrutturazione di immobili; i secondi, di importo generalmente inferiore, oscillante fra un massimo di 5 o 10 milioni in dipendenza dell'Istituto di Credito a cui ci si rivolge, vengono concessi a fronte di esigenze correnti (cura, studio, acquisto autovettura, arredamento, ecc.) e solitamente sono assistiti da garanzie personali o rilasciati in bianco.

Caratteristica comune di entrambi i finanziamenti è il rimborso a rate costanti altresì chiamato, matematicamente, ammortamento francese.

Per definizione l'ammortamento di un prestito, o rimborso graduale, si ha quando il debitore paga periodicamente, oltre agli interessi, anche una parte del capitale; la somma complessiva degli interessi del periodo, più il capitale rimborsato, si chiama rata.

Vi sono diversi tipi di ammortamento di un prestito: quello più in uso e che verrà preso in esame è detto a rate costanti o francese.

Un'altra caratteristica di questo ammortamento riguarda la composizione della rata, inizialmente composta quasi totalmente da interessi ed in minima parte dal capitale rimborsato. Per effetto del rimborso di quest'ultimo, la quota interessi diminuisce lasciando posto ad una maggior quota capitale, fino ad arrivare alle ultime rate nelle quali i rapporti iniziali vengono invertiti.

Il programma presentato permette di ottenere i suddetti piani di ammortamento per capitali interi fino a L. 999.999.999.=. Questa limitazione di importo è dettata da ragioni grafiche: cifre superiori non permetterebbero la stampa del piano su tre colonne così com'è strutturato.

Un'altra limitazione, sempre per necessità grafiche, è il numero delle rate in un massimo di 99. Si fa presente che il programma accetta un numero minimo di 2 rate perchè il rimborso in unica soluzione esula dalla definizione di ammortamento data in precedenza.

Dopo aver caricato il listato in figura 1, dare il RUN ed apparirà il menu.

Il programma principale si basa sulle istruzioni delle richieste n. 1 - Introduzione dati e n. 2 - piano di ammortamento.

Nella prima fase lo ZX81 chiede l'inserimento dei dati (capitale, tasso, numero rate e periodicità delle stesse) e successivamente calcola il piano per intero. Nella seconda richiesta provvede invece alla stampa dello stesso. Le richieste n. 3 - 4 - 5 sono parti del programma principale e permettono di ottenere dati senza attendere la stampa del piano completo.

Le REM inserite nel listato rendono chiara la lettura dello stesso.

Figura 1. *Il listato BASIC.*

```
 100 REM [illegible]
 110 REM [illegible]
 120 LET A$="[illegible]
[illegible]"
 130 DIM D$(32)
 140 LET R=0
 150 LET F$="ESTINTO"
 160 LET G$="RESIDUO"
 300 REM [illegible]
 310 CLS
 320 PRINT TAB 8;"*** M E N U **
*"
 330 PRINT ,,,,"1) INTRODUZIONE
DATI"
 340 PRINT ,,"2) PIANO DI AMMORT
AMENTO"
 350 PRINT ,,"3) COMPOSIZIONE RA
TA QUALUNQUE"
 360 PRINT ,,"4) RICERCA DEBITO
";F$
 370 PRINT ,,"5) RICERCA DEBITO
";G$
 380 PRINT ,,"6) REGISTRAZIONE D
ATI"
 390 INPUT A
 400 IF A<1 OR A>6 OR INT A<>A T
HEN GOTO 390
 410 CLS
 420 GOTO 1000*A
 500 REM [illegible]
 510 PRINT AT 21,0;"VUOI STAMPAR
E IL VIDEO?"
 520 INPUT E$
 530 IF CODE E$=51 THEN GOTO 560
 540 PRINT AT 21,0;D$
 550 COPY
 560 CLS
 570 RETURN
 600 LET M$=STR$ M
 610 IF LEN M$=9 THEN RETURN
 620 LET M$=" "+M$
 630 GOTO 610
 650 PRINT AT 20,0;"NON HAI INSE
RITO I DATI"
 660 PRINT "PREMI N/L PER TORNAR
E AL MENU"
 670 INPUT E$
 680 GOTO 300
 700 PRINT AT 20,0;"DOPO IL PAGA
MENTO DI QUALE RATA VUOI CONOSCE
RE IL ";
 710 IF A=4 THEN PRINT F$;"?"
 720 IF A=5 THEN PRINT G$;"?"
 730 INPUT X
 740 IF X<1 OR X>N OR INT X<>X T
HEN GOTO 730
 750 PRINT AT 20,0;D$+D$
 760 RETURN
 800 PRINT AT 0,0;A$;"CAPITALE:"
;C;" TASSO:";TR;"%"
 810 PRINT "RIMBORSO OGNI ";12/P
;" MESI CON N. ";N
 820 PRINT "RATE DI CUI ";N-1;"
RATE £ ";R
 830 PRINT "ED ULTIMA £ ";UR
 840 PRINT A$
 850 RETURN
1000 REM [illegible]
1010 PRINT AT 0,0;"CAPITALE: ";
1020 INPUT E$
1030 IF LEN E$<1 OR LEN E$>9 OR
INT VAL E$<>VAL E$ THEN GOTO 102
0
1040 LET C=VAL E$
1050 LET CR=C
1060 PRINT C
1070 PRINT ,,"TASSO: ";
1080 INPUT TR
1090 IF TR<=0 OR TR>100 THEN GOT
O 1080
1100 PRINT TR;"%"
1110 PRINT ,,"NUMERO RATE: ";
1120 INPUT N
1130 IF N<2 OR N>99 THEN GOTO 11
20
1140 PRINT N
1150 PRINT ,,"PERIODICITA' RATA:
"," 1 = ANNUALE",," 2 = SEMESTRA
LE",," 3 = QUADRIMESTRALE"," 4 =
 TRIMESTRALE"," 6 = BIMESTRALE",
,"12 = MENSILE"
1160 INPUT P
1170 IF P<1 OR P>12 OR P=5 OR P>
7 AND P<>12 THEN GOTO 1160
1180 PRINT AT 6,19;P
1190 PRINT AT 21,0;"SONO ESATTI
I DATI?"
1200 INPUT E$
1210 CLS
1220 IF CODE E$=51 THEN GOTO 101
0
1230 FAST
1240 LET T=TR/P/100
1250 LET R=((1+T)**N*T)/((1+T)**
N-1)*C
1260 LET R=INT R+1
1500 DIM C$(N,9)
1510 DIM C$(N,9)
1520 DIM I$(N,9)
1530 DIM R$(N,9)
1540 DIM N$(N,2)
1550 LET TI=0
```



OLTRE L'ORIZZONTE
CON LO SPECTRUM
SCONTO 20% AGLI ABBONATI
FINO AL 28-2-'84
77 PROGRAMMI PER SPECTRUM
GRAFICA - BUSINESS GRAFICA - UTILITY - ANIMAZIONI - MUSICA - GIOCHI
di Gaetano Marano
77 PROGRAMMI
PER SPECTRUM
Pagine. zioni a colori.
Cod.
L. 16000
E PER ZX81...
PROGRAMMI PER ZX81
E ZX80 CON NUOVA ROM
+ HARDWARE
144 Pagine
Cod. 520 D
L. 12000
PROGRAMMI
PER ZX81
CON NUOVA ROM
+HARDWARE
Per ordinare il vol me utilizzare l'apposito taglian o inserito in fondo alla rivista

PERSONAL SOFTWARE

Piani di ammortamento mutui con lo ZX81

Seguito figura 1.

```
1560 FOR I=1 TO N
1570 LET F=CR*T
1580 IF INT F=INT (F+.5) THEN LE
T M=INT F
1590 IF INT F<INT (F+.5) THEN LE
T M=INT F+1
1600 GOSUB 600
1610 LET I$(I)=M$
1620 LET TI=TI+M
1630 LET M=R-M
1640 IF I=N THEN LET M=CR
1650 GOSUB 600
1660 LET C$(I)=M$
1670 LET M=CR-M
1680 LET CR=M
1690 GOSUB 600
1695 LET R$(I)=M$
1700 LET M$=STR$ I
1710 IF LEN M$=1 THEN LET M$=" "
+M$
1720 LET N$(I)=M$
1730 NEXT I
1740 LET UR=VAL I$(N)+VAL C$(N)
1750 SLOW
1760 GOTO 300
2000 REM [illegible]
2010 IF R=0 THEN GOTO 650
2020 GOSUB 600
2030 PRINT " █ QUOTA █ QUOTA
 █ CAPITALE █INTERESSI█CAPITAL
E █ RESIDUO ";A$
2040 FOR I=1 TO N
2050 PRINT N$(I);"█";I$(I);"█";C
$(I);"█";R$(I)
```

Si è evitato l'uso della funzione INKEY$ per permettere l'utilizzo del programma anche sullo ZX80 nuova ROM senza slow, senza alcuna modifica.

La linea 6090 controlla, se nella zona Vars sono presenti delle variabili e quindi se il programma ha già girato.

In questo modo è possibile registrare il programma appena battuto con l'istruzione GOTO 6040, infatti non vengono saltate le prime linee di definizione variabili.

Il programma senza dati occupa 4 Kbyte di memoria; con un piano di ammortamento di 99 rate ne occupa quasi 8. ■

Seguito figura 1.

```
2060 IF I=11 OR I=32 OR I=53 OR
I=74 OR I=95 OR I=N THEN GOSUB 5
00
2070 NEXT I
2080 PRINT A$;"█";TAB 31;"█ TOT
ALE INTERESSI: ";TI;TAB 31;"█";
TAB 31;"█";A$
2090 GOSUB 500
2100 GOTO 300
3000 REM [illegible]
3010 IF R=0 THEN GOTO 650
3020 GOSUB 800
3030 PRINT AT 20,0;"DI QUALE RAT
A VUOI CONOSCERE LA COMPOSIZIONE
?"
3040 INPUT NR
3050 IF NR<1 OR NR>N OR INT NR<>
NR THEN GOTO 3040
3060 PRINT AT 20,0;D$+D$
3070 PRINT AT 7,0;"LA RATA N. ";
NR;" E" COMPOSTA DA:"
3080 PRINT "- QUOTA INTERESSI: "
;I$(NR)
3090 PRINT "- QUOTA CAPITALE : "
;C$(NR)
3100 PRINT A$
3110 GOSUB 500
3120 PRINT AT 21,0;"ALTRA RATA?"
3130 INPUT E$
3140 IF CODE E$=51 THEN GOTO 300
3150 PRINT AT 21,0;D$
3160 GOTO 3020
4000 REM [illegible]
4010 IF R=0 THEN GOTO 650
4020 GOSUB 800
4030 GOSUB 700
4040 PRINT AT 7,0;"DOPO LA RATA
N. ";X;" IL DEBITO"
4050 PRINT F$;" AMMONTA A £ ";C-
VAL R$(X)
4060 PRINT A$
4070 GOSUB 500
4080 PRINT AT 21,0;"ALTRA RICHIE
STA?"
4090 INPUT E$
4100 IF CODE E$=51 THEN GOTO 300
4110 PRINT AT 21,0;D$
4120 GOTO 4020
5000 REM [illegible]
5010 IF R=0 THEN GOTO 650
5020 GOSUB 600
5030 GOSUB 700
5040 PRINT AT 7,0;"DOPO LA RATA
N. ";X;" IL DEBITO"
5050 PRINT G$;" AMMONTA A £ ";R$
(X)
5060 PRINT A$
5070 GOSUB 500
5080 PRINT AT 21,0;"ALTRA RICHIE
STA?"
5090 INPUT E$
5100 IF CODE E$=51 THEN GOTO 300
5110 PRINT AT 21,0;D$
5120 GOTO 5020
6000 REM [illegible]
6010 PRINT AT 11,5;"VUOI REGISTR
ARE I DATI?"
6020 INPUT E$
6030 IF CODE E$=51 THEN GOTO 300
6040 PRINT ,,"FAI PARTIRE IL REG
ISTRATORE E POI PREMI NEW LINE
"
6050 INPUT E$
6060 FOR I=1 TO 50
6070 NEXT I
6080 SAVE "AMMORTAMENT█"
6090 IF PEEK (PEEK 16400+256*PEE
K 16401)=128 THEN RUN
6100 GOTO 300
```

C	Capitale mutuato.
TR	Tasso percentuale.
N	Numero rate.
P	Periodità pagamento rata.
T	Tasso del periodo.
R	Importo rate.
UR	Importo ultima rata.
C$(N,9)	Matrice contenente l'importo del capitale rimborsato ad ogni rata.
I$(N,9)	Matrice contenente l'importo degli interessi di ogni rata.
R$(N,9)	Matrice contenente il residuo debito dopo il pagamento della rata.
N$(N,9)	Matrice contenente il numero delle rate necessarie per l'incolonnamento.
TI	Totale degli interessi pagati.
D$	Maschera per la cancellazione del video.

Figura 2. *Descrizione delle variabili.*

Un esempio di analisi lessicografica

L'analisi dei testi con ZX Spectrum

di *Francesco Sardo*

L'analisi filologica o lessicografica di un testo consiste nella scomposizione del testo stesso per accertare la frequenza di comparizione di ogni parola.

L'analisi filologica è fondamentale strumento critico: l'ampiezza del lessico di un autore e la sua consistenza sono elementi essenziali per studiarne e valutarne l'opera.

Ad esempio, per scrivere *Le avventure di Pinocchio*, Collodi usò 6.000 parole diverse, di cui 3.300 una volta soltanto. Le parole più usate sono *burattino, povero, casa, strada*.

Giorgio Manganelli, sul *Corriere della sera*, ricava da questi dati gli elementi per un'analisi critica dell'opera e dello scrittore.

Inoltre, l'analisi lessicografica è uno dei sistemi più efficaci per accertare l'autenticità di un documento, cioè la sua reale appartenenza ad un autore. Confrontando infatti la frequenza delle parole rilevate in un testo in esame con quelle di un'opera certamente appartenente all'autore (possibilmente riguardante lo stesso argomento), si può accertare con un buon margine di sicurezza l'autenticità del testo in esame.

Questo tipo di esame viene effettuato abitualmente con grossi mainframe ma possiamo provare a fare qualcosa di analogo con uno ZX Spectrum con 48 kbyte di memoria.

Naturalmente il testo da esaminare non dovrà essere troppo lungo, in generale non più di 4-5 pagine.

Il programma fornisce alla fine l'elenco delle parole usate in ordine decrescente di frequenza, col numero di volte in cui sono state ripetute, e il numero totale di parole componenti il testo esaminato.

Per comodità, il testo viene introdotto frase per frase. Per segnalare la fine della fase di input, occorrerà introdurre come ultima stringa un asterisco.

Come possibile modifica al programma, si segnala quella di ordinare alfabeticamente le parole usate (e raccolte nel vettore stringa A $): basta che nella routine di riordinamento si operi sugli elementi di A $ invece che su quelli di N (vettore numerico contenente il numero di ripetizioni). A tale scopo basta modificare la linea 2130.

Le parole del testo vengono collocate nel vettore col metodo Hashing.

Il vettore A $ quindi non dovrà essere riempito per più dell'80%.

Ciò significa che le parole usate non dovranno essere più di 800, anche se potranno essere ripetute quante volte si vuole.

Descrizione del programma

Le linee 10/30 dimensionano i vettori e inizializzano le variabili; le linee da 40 a 60 e la linea 80 scompongono il testo in parole, mandando ogni parola alla subroutine 1000 per la sua collocazione in un elemento del vettore A $.

La linea 70 rimanda all'input di una nuova stringa una volta ultimata l'analisi della stringa introdotta precedentemente.

Le linee da 100 a 260 ordinano il vettore A $ in ordine decrescente di frequenza, e lo stampano, assieme alla frequenza di comparizione e al numero totale di parole esaminate.

L'ordinamento del vettore A $ è stato effettuato con il metodo di Shell-Metzner.

```
  10 REM analisi lessicografica
  11 REM ****** dei testi ******
  12 REM ***********************
  15 REM  scomposizione
  20 DIM A$(1000,15): DIM N(1000
)
  21 PRINT "Introdurre un perioc
o alla voltaconcludendolo con un
 segno di interpunzione."
  22 PRINT "finito l'input,dare
un asterisco"
  25 LET I=0
  30 LET J=1: LET K=1
  40 INPUT Z$: IF Z$="*" THEN GO
 TO 100
  50 IF Z$(K)=" " OR Z$(K)="," O
R Z$(K)="." OR Z$(K)=";" OR Z$(K
)=":" OR Z$(K)="'" THEN GO SUB 1
000
  60 LET K=K+1
  70 IF K>LEN Z$ THEN GO TO 30
  80 GO TO 50
 100 REM ordinamento e stampa
 105 PRINT AT 10,7; FLASH 1;"ATT
ENDERE,PREGO"
 110 GO SUB 2000
 120 CLS
 190 PRINT "parole:";I
 195 PRINT
 200 FOR G=1 TO 1000
 210 IF A$(G)="               "
THEN GO TO 250
 220 PRINT N(G),A$(G)
 250 NEXT G
 260 STOP
 999 REM ***********************
1000 REM smistamento hashing
1010 LET I=I+1
1020 LET B$=Z$(J TO K-1)
1030 LET J=K+1: LET K=K+1
1070 LET H=0
1080 FOR D=1 TO LEN B$
1090 LET H=H+CODE b$(D)
1100 NEXT D
1110 FOR E=LEN B$+1 TO 15
1120 LET B$=B$+" "
1130 NEXT E
1180 IF H>1000 THEN LET H=1000
1200 IF H=0 THEN LET H=500
1220 IF A$(H)="               "
OR A$(H)=B$ THEN GO TO 1260
1230 LET H=H-1
1250 GO TO 1200
1260 LET A$(H)=B$: LET N(H)=N(H)
+1
1270 RETURN
1999 REM ***********************
2000 REM sort di Shell-Metzner
2010 LET N=1000
2020 LET F=N
2070 LET F=INT (F/2)
2080 IF F=0 THEN GO TO 2230
2090 LET R=N-F
2100 LET B=1
2110 LET A=B
2120 LET S=A+F
2130 IF N(A)<N(S) THEN GO TO 217
0
2140 LET B=B+1
2150 IF B>R THEN GO TO 2070
2160 GO TO 2110
2170 LET T=N(A): LET T$=A$(A)
2180 LET N(A)=N(S): LET A$(A)=A$
(S)
2190 LET N(S)=T: LET A$(S)=T$
2200 LET A=A-F
2210 IF A<1 THEN GO TO 2140
2220 GO TO 2120
2230 RETURN
```

Listato 1. *Il programma di analisi testi.*

La routine 1000 colloca al loro posto le singole parole col metodo Hashing, ricavando dal codice delle lettere che le compongono l'indirizzo ove ubicarle. Ogni volta che in un elemento del vettore viene collocata una parola, il corrispondente elemento del vettore numerico N viene aumentato di 1.

È importante notare che bisogna chiudere ogni frase introdotta con un segno di interpunzione.

Le frasi potranno essere lunghe quanto si vuole; per comodità di battitura e per eventuali correzioni, è bene utilizzare una nuova stringa per ogni periodo. ■

Per ordinare il volume utilizzare l'apposito tagliando inserito in fondo alla rivista.

Z-80
Pag. 530 **L. 26.000**
Cod. 328D Formato 14,5 x 21

Questi due libri sono stati ideati come testi autonomi e completi per imparare la programmazione in linguaggio Assembler, usando lo Z80 o il 6502 (i microprocessori forse più diffusi).
Scorrevoli da leggere, non richiedono alcuna conoscenza di base, ne di elettronica generale né di programmazione.
Sono stati progettati, infatti, sotto forma di corso che, sistematicamente, passo dopo passo, porta il lettore dai concetti di base fino alle tecniche di programmazione avanzate, al fine di permettergli la realizzazione di programmi sempre più complessi.
L'esposizione progressiva, rigorosamente strutturata, comporta la risoluzione obbligatoria di esercizi attentamente graduati al fine di verificare che si sia veramente capito quanto presentato?
Ben si prestano, perciò, a chi si avvicina per la prima volta ai microprocessori e ne vuole conoscere e capire gli aspetti essenziali di programmazione. Per tutti coloro che già hanno programmato, invece, sarà una vera e propria miniera di informazioni sulle caratteristiche specifiche del microprocessore d'interesse, evidenziandone, nel contempo, vantaggi e svantaggi.



6502
Pag. 390
L. 25.000
Cod. 503B
Formato 14,5 x 21

La Potenza dei Microprocessori

SCONTO 20% AGLI ABBONATI FINO AL 28-2-'84

SOMMARIO
Concetti Fondamentali; Organizzazione Hardware del Microprocessore; Tecniche Fondamentali di Programmazione; Set di Istruzioni; Tecniche di Indirizzamento; Tecniche di Input/Output; Dispositivi di Input/Output; Esempi Applicativi; Strutture dei Dati; Sviluppo del Programma; Conclusioni.

GRUPPO EDITORIALE JACKSON
Divisione Libri

PERSONAL SOFTWARE

Alcuni trucchi dello Spectrum

Spectrum in guardia, non hai più segreti ...

di *Tullio Policastro*

Questo articolo è una collezione per ZX Spectrum di quei piccoli accorgimenti di programmazione (e non) che si scoprono col tempo e con le prove, ma soprattutto collezionando le idee più brillanti di diversi libri e riviste specializzate. Senza indugiare ulteriormente, eccone una decina, e chi più ne ha più ne metta.

1) Capita spesso durante la stesura o la correzione di un programma di voler fermare l'esecuzione in corrispondenza di una certa richiesta di input. Ci sono tre possibilità:

- Se l'input è numerico (non compaiono le virgolette) basta rispondere con STOP (simbol shift + A), oppure causare un errore fornendo il nome di una variabile inesistente.
- Se viene richiesta una stringa (e vengono stampate le virgolette) occorre cancellare almeno la prima virgoletta e poi usare STOP come per gli input numerici. Per cancellare le virgolette si può usare EDIT (caps shift + 1) o DELETE (caps shift + 0).
- Se viene richiesta una stringa con l'opzione LINE non vengono stampate le virgolette e non è quindi possibile cancellarle ed usare STOP. In questo caso occorre usare la freccia verso il basso (caps shift + 6) per fermare il programma.

Ovviamente in nessuno di questi casi funziona il tasto BREAK.

2) Quando, si desidera cancellare tutto quel che si è scritto di un'istruzione, magari perchè contiene un errore che non si riesce a correggere basta premere EDIT (CAPS SHIFT + 1), seguito da ENTER (in pratica l'istruzione corrente scende al posto di quella che si stava scrivendo, poi torna al suo posto invariata).

3) L'effetto di bloccare la stampa per consentire la lettura dello schermo, e di riprenderla semplicemente premendo un tasto qualsiasi (*senza* ENTER, ma incluso il singolo tasto ENTER), che in certi programmi vediamo segnalato con il messaggio "Per seguitare, premi un tasto" ("Press a key to continue"), si ottiene inserendo pause nel punto opportuno del programma.

Si ricordi che anche le pause di lunghezza programmata (in 1/50 di secondo) tipo PAUSE *n* vengono interrotte, ed il programma riprende dal punto immediatamente dopo, premendo un tasto qualsiasi, dopo l'inizio della pausa.

4) Lo Spectrum è dotato di un segnale acustico, spesso poco udibile, che consente di avvertire la avvenuta pressione di un tasto. Una segnalazione acustica migliore, si ottiene con POKE 23609, *n*, con *n* variabile a piacere fra 5 e 250. Questa istruzione può essere utilmente inserita all'inizio di un programma.

5) Il CAPS LOCK, ovvero la scrittura delle lettere dell'alfabeto in maiuscolo, si ottiene con comando diretto premendo CAPS SHIFT + 2 (se ripetuto, torna alla scrittura in minuscolo). Per avere la certezza che una certa stringa, che il programma vuole sia inserita in maiuscolo, sia realmente tale senza dover premere i tasti suddetti, si può usare POKE 23658,8 prima della input.

Il comando contrario è POKE 23658,0.

6) Quando si deve ripetere di seguito lo stesso carattere basta mantenere premuto il tasto per attivare la funzione "REPEAT" automatico se si desidera una maggiore velocità e la retrocessione del cursore lungo i caratteri della linea (magari per accellerare l'avanzamento in EDIT) si può intervenire sia sulla velocità di avanzamento che sul tempo di ritardo di intervento con due POKE: POKE 23561, *m* (con *m* es. inferiore a 35) e POKE 23562, *n* (con n minore di 5). Per tornare alle condizioni normali, si inseriscono con gli stessi POKE i limiti superiori indicati, oppure eseguire new il computer.

7) Molte volte, quando si devono fare delle prove sulla velocità relativa di esecuzione di diversi modi di programmazione, o di diversi programmi (ad esempio quelli per eseguire delle ricerche in un file, e simili) è utile disporre di un "contasecondi", da software. Il particolare meccanismo di temporizzazione dello Spectrum ed il contenuto di tre particolari celle di memoria (la variabile di sistema FRAMES) consentono di fare tale operazione con notevole accuratezza. Il meccanismo è spiegato nel capitolo 27 del manuale (si noti però che nel testo italiano c'è un piccolo errore: nel programmino che viene descritto, di tre sole istruzioni, manca un FNu(), che deve essere ripetuto due volte). Questo è il programmino esatto:

```
10 DEF  FNm(x,y)=(x+y+ABS(x-
   y))/2
20 DEF  FNu()=(65536 ★ PEEK
   23674+256 ★ PEEK 23673 ★
   PEEK
   23672)/50
30 DEF FNt()=FNm(FNu(),F-
   Nu())
```

Queste tre istruzioni (i numeri di riga possono ovviamente essere diversi) si possono inserire ovunque nel programma. Immediatamente *prima* della parte di programma di cui si vuole misurare la durata di esecuzione si inserisce (n° di riga)

KE 23674,0 ed immediatamente *dopo* il termine del programma di cui si vuole misurare la durata si pone un ... PRINT FNt(). Il valore che viene stampato o visualizzato è il tempo, in secondi, intercorso fra queste due istruzioni.

8) Il BASIC Sinclair si presta all'uso particolare della funzione logica AND in espressioni del tipo a$ AND *a*. Tale istruzione ritorna stringa nulla se *a* = 0 o stringa a$ se *a* è diverso da zero, si può utilizzare quindi per stampare due messaggi alternativi in corrispondenza a due situazioni, senza bisogno di impiegare IF...THEN. Se a$ a b$ sono i due messaggi, basta trovare un'espressione da sostituire a *c* (nell'istruzione che segue) e che assuma in un caso il valore zero, nell'altro un valore diverso da zero:

PRINT a$ AND *c* + b$ AND *c*

9) Talvolta si può desiderare che una certa parte del listato risulti "invisibile", perchè ad esempio non si vuole che altri possano vedere come è scritto il programma od una parte di esso. L'esistenza del colore bianco anche per l'INK permette di ottenere quasi interamente tale scopo, operando in questo modo:

- Mediante un LIST *n* (seguito da SPACE per impedire lo scroll) si porta il cursore davanti alla riga a partire dalla quale si vuole impedire la lettura del listato; con EDIT (CAPS SHIFT + 1) si fa scendere la riga stessa in calce al quadro.
- Si prende nota del numero della ultima riga della parte che si vuole rendere "invisibile".
- Si passa in modo "E", premendo entrambi i tasti di SHIFT allo stesso tempo; quindi si preme CAPS SHIFT+7 (equivale ad INK 7: se il colore del fondo (PAPER) fosse diverso, ossia non bianco, si sostituisca il relativo numero del colore al posto del 7), ed infine ENTER: la riga torna al suo posto; si nota che *tutto* il resto del programma è sparito, eccetto il numero della riga.
- Si esegua LIST numero dell'ultima riga da "occultare" + ENTER e poi EDIT (CAPS SHIFT+1). La riga in questione apparirà al fondo del quadro. (Quindi il listato è quasi invisibile: infatti se si richiama con le funzioni di editing, ossia le frecce e l'EDIT, una qualsiasi istruzione "invisibile", questa diventa visibile in fondo al quadro; il listato occulto si può quindi leggere una riga per volta).
- Andate con la "freccia a destra" (CAPS SHIFT+8) sino al termine della riga editata.
- Tornate in modo "E" (i due tasti SHIFT insieme), e infine premete CAPS SHIFT+0 (equivale a INK 0: variare se del caso) e poi ENTER: le righe del listato successive all'ultima da occultare torneranno visibili.

Se si vuole rendere "invisibile" in una sola volta tutto il programma, esiste un altro sistema più semplice:

- Impostate una istruzione n° 1 come segue: 1/INV VIDEO (=CAPS SHIFT+4)/PRINT/TRUE VIDEO (=CAPS SHIFT+3)/"/"/ ENTER (senza le barre di separazione, ovviamente).
- Eseguite EDIT (CAPS SHIFT+1) e poi: TRUE VIDEO (= CAPS SHIFT+3)/CAPS SHIFT+8/INV VIDEO (=CAPS SHIFT+4)/ENTER (il ronzio che sentirete a un certo punto, che normalmente significa memoria piena, qui è regolare). Il listato si riduce ad un "1". Il 231 di BORDER, o il 234 di REM (notate che ho citato codici di comandi, perchè quando il tasto verrà azionato solitamente si è in modo "K" ed il significato del tasto premuto è quello del comando inscritto).

Il programma eseguirà allora un sottoprogramma in l.m. e, automaticamente, il NEW. Ora si imposterà una riga:

1 PRINT "Questo è l'esempio" seguita da ENTER;

poi EDIT (CAPS SHIFT+1), e a questo punto si inserirà immediatamente dopo l'1 del n° di riga REM (prima del PRINT, nell'esempio), e si premerà ENTER. Infine si renderà operativo il tutto con RANDOMIZE USR 65110 + ENTER. (Per il 16 Kbyte, 32329 invece di 65110, ossia a + 11).

Quando si premerà, in modo "K", il tasto prescelto, la riga 1 apparirà per un istante al fondo del quadro e verrà *eseguita* (nell'1 REM impostata come detto sopra, si inserirà il comando che interessa).

Quando si vorrà annullare l'effetto del "tasto controllo" così reso attivo, si imposterà RANDOMIZE USR 65100 + ENTER (32319 per il 16 Kbyte). (Ricordarsi di farlo al termine dell'uso).

L'operazione è particolarmente utile per stampare a richiesta il numero di byte disponibili in memoria in qualsiasi istante. L'istruzione 1 assume allora la forma:

1 REM PRINT "Byte liberi:"; 65365 - (PEEK 23653 + 256★ PEEK 23654)

In realtà non sono disponibili, di questi, un limitato numero di byte posti nel "machine stack" e nel "GOSUB stack"; per una lettura corretta occorre ricorrere ancora al l.m., che permette di leggere l'indirizzo dello "stack pointer". Memorizzando l'idoneo programmino di pochi byte *prima* del l.m. fornito dai DATA di riga 20, ossia con:

10 CLEAR a-14 (65085 per il 48 Kbyte; 32304 per 16 Kbyte)
15 DATA 33, 0, 0, 57, 237, 91, 101, 92, 167, 237, 82, 229, 193, 201
....

e partendo da 65086 (=a—13; 32305 per il 16 Kbyte) come valore iniziale per il ciclo FOR ... NEXT di riga 50, si realizzano le modifiche necessarie. La riga 1 assume allora la forma:

1 REM PRINT "Byte liberi"; USR 65086 (per 48/16 Kbyte: 65086, risp. 32305).

In questo modo è possibile ottenere un solo tasto controllo definito dall'utente per volta.

Inoltre, questo metodo si presta solo per comandi (istruzioni) non troppo complessi, preferibilmente "monolinea"; e per stampare messaggi non più lunghi di 1 riga (32 caratteri).

10) Per sapere come sono formati i

Alcuni trucchi dello Spectrum



vari caratteri stampabili dello Spectrum, si può utilizzare il seguente programmino (che va a leggere il contenuto a 18 byte) a partire da 256 locazioni più avanti di quella (normalm. 15616) dalla variabile di sistema CHARS, (posta nei due byte 23606/7) e visualizza la matrice di punti con spazi bianchi e/o inversi. ■

Listato 1. *Il listato BASIC.*

```
  10 INPUT "Carattere";a$: CLS :
 IF a$="" THEN GO TO 200
  20 LET c=CODE a$-32
  30 FOR i=0 TO 7: LET n=PEEK (1
5616+i+8*c)
  40 DIM b(8): LET a=128
  50 FOR j=1 TO 8: IF n<a THEN G
O TO 70
  60 LET b(j)=1: LET n=n-a: IF n
=0 THEN GO TO 80
  70 LET a=a/2: NEXT j
  80 FOR j=1 TO 8: LET a$="": IF
 b(j)=1 THEN LET a$="█"
  90 PRINT AT 9+i,10+j;a$: NEXT
j
 100 NEXT i: GO TO 10
```

COMMODORE VIC 20/C 64

Disabilitazione List: la nuova funzione di Append

di Alessandro Guida

A volte è utile poter disabilitare alcune funzioni, si pensi ad un programma dimostrativo che non si vuole venga fermato o listato.

Vedremo come è possibile ottenere ciò con una sola istruzione POKE.

Questo mese abbiamo anche la collaborazione di un lettore, il Sig. Lucio Iacono che illustra un procedimento per poter accodare ad un programma,un altro caricato da nastro o disco.

Disabilitazione funzioni sul VIC e C 64

Nella tabella 1 sono riportati i valori da introdurre nelle opportune locazioni per disabilitare alcune funzioni del computer.

La colonna centrale riporta i valori per la disabilitazione, mentre quella a destra riporta i valori normali. Queste istruzioni di POKE possono essere inserite nei programmi o date in modo diretto.

1) *Disabilitazione STOP RESTORE e LIST.* Il trucco per disabilitare lo STOP è quello di cambiare il contenuto del vettore di Test-STOP $0328, $0329. Questo vettore contiene l'indirizzo di una subroutine che controlla se questo tasto è stato premuto. Il rimedio è quindi cambiare l'indirizzo di partenza in modo che non venga più effettuato il controllo. Questa modifica va fatta tenendo presente che al ritorno dalla subroutine di Test-STOP l'accumulatore non deve contenere 0 e il registro Y deve rimanere invariato. Con il POKE suggerito nella tabella 1 otteniamo la prima parte, ma il registro Y verrà modificato.

Qual'è il risultato? È presto detto. Poichè all'interno della routine di LIST è chiamata la subroutine di Test-STOP avremo che, essendo cambiato il registro Y, che serve da puntatore nella linea Basic da listare, il listato si tradurrà in una sequenza di caratteri insignificanti. A questo va aggiunto che il tasto di RESTORE per espletare la sua funzione ha bisogno di essere premuto insieme a quello di STOP. Perciò anche il RESTORE resterà inibito.

2) *Disabilitazione RESTORE.* Anche per il tasto RESTORE esiste un vettore che contiene l'indirizzo della routine da eseguire nel caso questo venga premuto. Questo vettore è il $0318, $0319 nel quale metteremo l'indirizzo di una locazione contenente una istruzione di ritorno RTI.

Funzione disabil.	Disabilitazione	Riabilitazione
1) STOP, RESTORE LIST	POKE 808,100 POKE 808,225	POKE 808,112 (VIC 20) POKE 808,237 (C 64)
2) RESTORE	POKE 792,90 POKE 793,203	POKE 792,173 (VIC 20) POKE 793,240
3) SAVE	POKE 818,PEEK (816) POKE 819,PEEK (817)	(VIC 20 e C 64)
4) LIST	POKE 775,0 POKE 775,0	POKE 775,119 (VIC 20) POKE 775,167 (C 64)

Tabella 1. *I valori necessari per la disabilitazione di alcune funzioni sul VIC20 e sul C64.*

3) *Disabilitazione SAVE.* Altri due interessanti vettori nel VIC (o 64) sono quelli che contengono gli indirizzi delle routine di LOAD e SAVE. Rispettivamente $0330, $0331, e $0332, $0333. È evidente che mettendo nel vettore di SAVE l'indirizzo della routine di LOAD la prima funzione diventerà inutilizzabile.

4) *Disabilitazione LIST.* Per finire vediamo l'ultimo POKE che blocca del tutto l'istruzione di LIST. Per capirne il funzionamento ricordiamo che le linee basic sono conservate in maniera codificata nella memoria del computer. Infatti, ogni parola chiave (comando) del BASIC è tradotto in un codice di un solo byte. Perciò la routine di LIST necessita, a sua volta, di una subroutine che ritraduca questi codici in comandi BASIC. Inutile dire che l'indirizzo di partenza di questa routine è modificabile, in quanto conservato nel vettore $0306, $0307. Con il valore in tabella si avrà un RESTORE ogni volta che si tenterà di effettuare un listato.

A questo punto un comando di LIST ci mostrerà lo schermo vuoto, ma niente paura, il programma è solo nascosto sotto il nuovo principio della memoria. Sarà ora possibile caricare il programma (coda) dal nastro con il solito LOAD "Nome" oppure solamente LOAD.

Caricato il programma e verificato che giri regolarmente basterà battere:

POKE 43,1 : POKE 44,16

per riportare la memoria alle dimensioni iniziali.

Al comando LIST apparirà ora tutto il programma completo delle due parti, come se fosse stato scritto tutto insieme.

Questo procedimento può essere ripetuto anche più volte avendo l'accortezza di usare per ogni pezzo che si aggiunge numeri di linea sempre maggiori di quelli esistenti.

Il programma così composto potrà essere usato e salvato come un qualunque altro programma.

È appena il caso di ricordare che usando le espansioni di memoria occorre modificare conseguentemente le POKE 43 e 44 dell'ultimo comando.

A questo scopo, per evitare errori basta battere una:
PRINT PEEK (43), PEEK (44)
prima di cominciare e prendere nota della risposta che andrà poi inserita nelle ultime POKE per ripristinare le condizioni iniziali.

Se avete scoperto qualche piccolo segreto del VIC 20 (o del C64) o se avete qualche trucchetto per facilitare la programmazione comunicatelo agli altri lettori, inviando un articolo al seguente indirizzo:
Alessandro Guida - c/o **Personal Software** - Via Rosellini, 12 - 20124 Milano

COMMODORE VIC 20

Come realizzare la funzione Append

di Lucio Iacono

La lettura dell'interessantissimo articolo a firma di Luciano Gemme sul n. 8/9 di **Personal Software** è stata come il classico colpo di fulmine: visto che tutto il trucco della memorizzazione delle istruzioni del VIC 20 è concentrato nei "link" e che questi possono essere letti e modificati con la massima facilità, e visto che è così facile recuperare un programma andato perso per un malaccorto comando NEW, mi sono chiesto se era possibile qualche altra manipolazione sui programmi residenti nella memoria del VIC 20.

Man mano che si scrivono i programmi il sistema operativo del VIC provvede a memorizzarli con una tecnica che sembra complessa ma in realtà è assai semplice. La prima istruzione viene memorizzata scrivendo nei primi due byte liberi della memoria (4097 e 4098) l'indirizzo di inizio della 2ª istruzione.

In tutti i suoi calcoli interni il VIC usa il primo byte per la parte bassa del numero e il secondo per la parte alta: per ricostruire il valore decimale basta fare:
PEEK(4097)+PEEK(4098) ★ 256.

Poi viene memorizzato il numero di linea, sempre con lo stesso sistema nei due byte successivi, poi il testo opportunamente compattato.

I comandi BASIC sono rappresentati con codici (chiamati TOKEN). Ad esempio il RETURN è codificato con uno 0.

Vengono poi posti a 0 i due byte immediatamente successivi per indicare la fine del programma. Contemporaneamente, ad ogni istruzione battuta, viene incrementato il contenuto dei due byte 45 e 46 che puntano l'indirizzo della prima cella di memoria libera dopo i due zeri di fine programma e quindi l'inizio dell'area di memorizzazione delle variabili.

Quando invece si carica un programma registrato su nastro il contenuto del nastro viene depositato a cominciare dalla locazione puntata dalle celle 43 e 44 che contengono invece l'inizio della memoria disponibile; questo spiega perchè caricando da nastro si perde qualunque programma già residente in memoria.

Però se si sposta il puntatore costituito dal contenuto delle celle 43 e 44 in modo che punti oltre la fine del programma già residente in memoria, le istruzioni riversate dal nastro si andranno a depositare in coda al programma esistente.

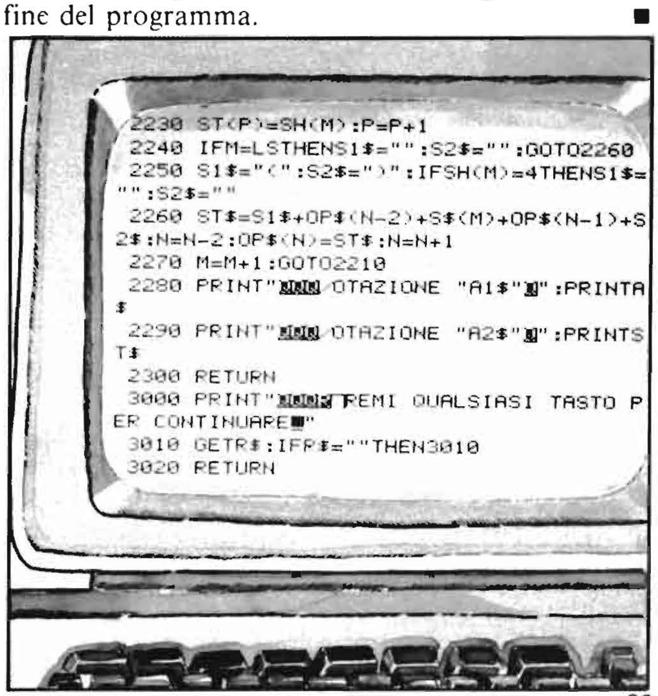
Ecco così realizzata la funzione "APPEND" presente solo sugli elaboratori più grossi, che consiste appunto nella possibilità di accodare, appendere un programma ad un altro già residente in memoria.

Vediamo ora come procedere, passo per passo.

Per prima cosa occorre digitare e salvare su nastro la parte terminale, la coda, del programma, badando che i numeri di linea siano tutti maggiori di quelli del programma cui dovrà essere appesa.

Verrà poi digitata la parte anteriore del programma, la testa, nel solito modo. Finita questa fase occorre spostare il limite di inizio della memoria all'altezza della fine del programma battendo:
POKE 43, PEEK(45)—2: POKE 44, PEEK (46).

Il — 2 serve per tenere conto dei due zeri aggiunti e per poter scrivere le nuove istruzioni sopra di essi, altrimenti la presenza dei due zeri segnalerebbe la fine del programma. ■



ZX SPECTRUM

Come proteggere i vostri programmi

di Marcello Spero

Molti programmi commerciali per lo Spectrum sono concepiti in modo da renderne impossibile l'interruzione del funzionamento, e quindi del programma. Trattandosi normalmente di programmi in linguaggio macchina è logico che il tasto BREAK non abbia effetto e che risultano quindi automaticamente protetti.

Meno ovvio è il loro comportamento durante il caricamento da cassetta: se, infatti, tentiamo di interromperlo premendo BREAK o lo spazio, otteniamo immediatamente un NEW, perdendo tutto quanto era già stato immagazzinato in memoria. In questo modo il programma è completamente "inespugnabile".

Ovviamente non possiamo conoscere con esattezza il metodo usato per ottenere questo effetto; esiste comunque un modo alla portata di tutti per rendere i programmi, anche scritti in BASIC, refrattari al BREAK.

La variabile di sistema ERR SP determina il comportamento della macchina in caso di errore. Essa, infatti, contiene l'indirizzo occupato all'interno della catasta di sistema (Machine Stack) dal puntatore alla routine di errore. Al verificarsi di un errore, segnalato dal cambiamento di valore della variabile ERR NR, il sistema salterà alla routine indicata, che produrrà un appropriato messaggio diagnostico. Tutto questo vale anche per il BREAK, che viene considerato un errore a tutti gli effetti.

Per modificare questo meccanismo occorre cambiare il valore del puntatore contenuto nella catasta; mettere le mani nella catasta del sistema è però un'operazione sconsigliabile: meglio fare in modo che ERR SP punti ad una coppia di byte, fuori dalla catasta, contenente l'indirizzo di una routine di nostra scelta.

Dobbiamo ora trovare una routine adatta ad essere eseguita in caso di errore. Volendo ottenere la scomparsa di tutto quanto c'è in memoria la scelta migliore è quella della routine che parte dall'indirizzo 0. Si tratta dell'inizializzazione del sistema, e viene eseguita ogni volta che accendiamo la macchina: opera un azzeramento di tutta la memoria, l'inizializzazione delle variabili di sistema e della grafica definibile.

Adottando questa soluzione in caso di errore otterremo il medesimo effetto di uno spegnimento e riaccensione del calcolatore.

Tutto ciò che dobbiamo fare è scegliere una coppia di locazioni cui far puntare ERR SP; la soluzione più semplice è utilizzare gli indirizzi 23728 e 23729, che costituiscono praticamente una variabile di sistema inutilizzata e contengono già il valore 0, che è appunto quello che ci serve.

Inserendo all'inizio di un programma le istruzioni

```
10 POKE 23613, 23728-256 ★ INT (23728/256)
20 POKE 23614, INT (23728/256)
```

(dove 23613 e 23614 sono gli indirizzi della variabile ERR SP) otteniamo un programma "che non perdona", ossia che in caso di errore si autodistrugge: attenzione quindi a proteggerlo con adeguate istruzioni di controllo da condizioni di errore che potrebbero verificarsi durante il suo funzionamento.

Tutto questo riguarda naturalmente i programmi BASIC poichè, come abbiamo detto, i programmi in linguaggio macchina sono per loro natura insensibili all'azione del BREAK ed immuni da condizioni di errore (o, piuttosto, non si accorgono degli eventuali errori in cui incorrono).

Come proteggere i vostri programmi

Il problema della protezione dall'interruzione si ripresenta tanto per il BASIC che per il linguaggio macchina nel caso del caricamento da cassetta; inutile, infatti, impedire il BREAK del programma in esecuzione, se è possibile listarlo prima che inizi a funzionare semplicemente interrompendone il caricamento pochi istanti prima che termini. L'unico modo di aggirare questo ostacolo è far sì che all'inizio del caricamento, cioè prima che il programma vero e proprio inizi ad essere caricato, la variabile ERR SP venga portata al valore che ci interessa. Procediamo perciò al salvataggio del programma non con:

SAVE "programma"

ma con:

SAVE "programma" CODE 23552,L

dove 23552 è l'indirizzo di inizio dell'area delle variabili di sistema, mentre il numero L di byte da salvare sarà dato da:
PEEK 23653+256 ★ PEEK 23654-23552
comprendendo in questo modo tutta la memoria fino all'inizio della zona libera, segnata dalla variabile STKEND, che si trova appunto all'indirizzo 23653. Il numero di byte da salvare potrebbe in realtà essere inferiore, ma è meglio essere prudenti; anzi, tenetevi abbondanti anche sulla cifra che otterrete nel modo visto sopra: meglio salvare qualche byte senza importanza che perderne uno essenziale.

Così facendo avremo all'inizio della nostra registrazione proprio le variabili di sistema. Per compiere tutta questa operazione, però, il programma deve essere fermo, e quindi ERR SP deve avere il suo valore originale; inoltre abbiamo bisogno di un auto start al termine del caricamento, o tutto questo lavoro sarebbe inutile. Soluzione:
POKE 23613, 23728 256 ★ INT(23728/256):
POKE 23614, INT(23728/256): SAVE "programma"
CODE 23552,L:GO TO 1
(attenzione: non RUN, che reinizializza ERR SP!) e il gioco è fatto.

Purtroppo non è possibile in questo caso l'uso di VERIFY per controllare la registrazione: infatti, appena terminato il salvataggio, il programma va in esecuzione e tentando di fermarlo con BREAK (ammesso che sia un programma BASIC) se ne provoca la scomparsa.

Caricando un programma in questo modo otteniamo altri vantaggi, legati al fatto che tutte le variabili di sistema assumono il valore che avevano al momento del salvataggio; otterremo quindi il ripristino automatico di determinati attributi, dell'eventuale "bip" di tastiera, e così via. Attenzione, invece, per la RAMTOP: anche qui si ha il ripristino automatico del valore in uso al momento del salvataggio, ma se questo è diverso da quello impostato al momento del caricamento il sistema rischia di saltare, e comunque ci si trova in una situazione di instabilità che conviene in ogni caso evitare. Per scongiurare situazioni di questo tipo i programmi che usano valori della RAMTOP diversi da quello normale dovrebbero avere come prima istruzione:
10 CLEAR
seguita da:
20 POKE 23613, 23728-256 ★ INT(23728/256)
30 POKE 23614, INT(23728/256)
poichè CLEAR, come RUN, reinizializza ERR SP.

In alternativa è possibile sostituire, nel comando multiplo di salvataggio visto prima, GO TO 1 con
:RUN: POKE 23613, 23728-256 ★ INT (23728/256):
POKE 23614, INT(23728/256)
che dà lo stesso effetto, ma rende veramente enorme il comando stesso.

Un'ultima considerazione da fare riguarda l'uso di questo metodo di protezione con i programmi BASIC. Oltre alle già CLEAR e RUN, infatti, anche altre istruzioni come ad esempio GO SUB E RETURN ripristinano il valore normale di ERR SP, rendendo necessario un uso intensivo di istruzioni POKE all'interno e dopo ciascuna subroutine per mantenere inalterata la protezione. In pratica questo pone dei limiti alla protezione di lunghi programmi BASIC, cui è più opportuno applicare una versione particolare di questa tecnica: il blocco del programma per mezzo di una parola chiave.

Il sistema è molto semplice: salvando con il sistema visto, per garantire la protezione durante il caricamento, un programma BASIC le cui prime istruzioni sono:
10 INPUT LINE a$
20 IF a$ <> "parola chiave" THEN GO TO 10
30 GO SUB 9999
cui farà riscontro, in linea 9999, una subroutine fittizia composta da un'unica linea:
9999 RETURN
(Questo per evitare due voluminose POKE; in alternativa, se il programma non contiene dati, cosa poco probabile visto che vogliamo proteggerlo, possiamo sostituire la linea 30 con l'istruzione):
30 CLEAR
In questo modo otteniamo una protezione durante il caricamento e fino al momento in cui l'utilizzatore avrà impostato la corretta parola chiave.

Superato questo punto il programma potrà essere interrotto e listato normalmente. In questo modo viene impedito l'accesso e la manomissione del programma da parte di persone non autorizzate.

SHARP

Tre nuove istruzioni

di Mauro Lenzi

Sperando di fare cosa gradita ai molti utilizzatori di pocket computer, cominciamo da questo numero la pubblicazione di una sezione fissa dedicata in particolare alle programmabili Sharp. Nei primi articoli il nostro collaboratore bolognese Mauro Lenzi si soffermerà sul modello PC-1251 introducendo anche alcune caratteristiche inedite di questa interessante macchina.

Ciò che mi propongo di fare, in questa rubrica, è di "vivisezionare" un computer, penetrando nei meandri più reconditi del suo sistema operativo.

Io tratterò principalmente del pocket-computer Sharp PC-1251, tuttavia i sistemi e le astuzie impiegati per svelarne i misteri, possono venire applicati alla maggior parte dei computer in commercio.

Le principali caratteristiche che occorrono per diventare dei buoni "computer-chirurghi" sono la pazienza e l'intuito: è infatti procedendo per tentativi che ho potuto scoprire *tre nuove istruzioni* presenti nel computer, ma non segnalate nei manuali di istruzioni!

Queste tre istruzioni consentono di Programmare il computer in linguaggio macchina!!

Vediamo di esaminarle una per volta.

Proviamo a digitare PEEK 10000, seguito naturalmente dal tasto enter: clamorosamente, invece di apparirci la scritta ERROR 1 come dovremmo aspettarci, compare il numero 39. Se facciamo altre prove, cambiando il numero che segue l'istruzione PEEK, sul display vengono mostrati altri numeri. Abbiamo appena scoperto che il nostro computer "riconosce" l'istruzione PEEK, che, pur essendo molto utile e conosciuta, non è segnalata sul manuale.

Scoperta l'istruzione PEEK, possiamo aspettarci che esista anche la corrispondente istruzione POKE; proviamo a digitare: POKE 50000, 92.

Il computer non segnala alcun errore. Come controprova digitiamo: PEEK 5000; sul visore appare il numero 92, che è appunto il numero che abbiamo precedentemente immesso nella locazione di memoria 50000 con il comando POKE.

Utilizzando esclusivamente queste due istruzioni in seguito faremo cose inimmaginabili, come per esempio fare comparire dei caratteri "cinesi", creare dei numeri di linea decimali oppure rendere "invisibili" alcune linee di un programma.

Con queste due istruzioni è inoltre possibile fare dei programmi in linguaggio macchina. Ma come è possibile farli eseguire al computer?

L'istruzione che viene utilizzata a questo scopo nel linguaggio BASIC si chiama CALL. Anche questa istruzione, come le due precedenti, in teoria non dovrebbe esserci, non essendo annotata sul manuale.

Digitiamo CALL 12: anche questa volta non ci perviene alcuna segnalazione di errore ed il computer la esegue; tuttavia, non essendoci niente di "interessante" nella locazione 12 e seguenti, il display non segnala niente di particolare.

Per coloro che desiderino avventurarsi nel difficile mondo del linguaggio macchina ho già detto abbastanza; adesso sta alla loro intraprendenza e alla loro pazienza proseguire le ricerche in questa direzione.

Invece consiglio a tutti coloro che non abbiamo già una discreta conoscenza di un linguaggio assembler o macchina, di "dimenticarsi" questa istruzione perchè procurerebbe soltanto fastidi. Se, per esempio dopo esserci assicurati di non avere in memoria alcun programma utile che desideriamo conservare, digitiamo CALL 16000, il computer va in "tilt". Sul display compare solo la scritta "busy", ma qualunque tentativo di fermarlo, perfino spegnendolo, si rivelerà inutile. A questo punto l'unico modo per farlo "rinsavire" è di RESETtarlo, ciò comporta l'azzeramento di tutta la memoria RAM.

La scoperta di queste istruzioni ci ha ormai aperto la via per accedere al sistema operativo del computer, la prossima volta scopriremo come viene organizzata la memoria ed altre pseudo-istruzioni misteriose. ■

TEXAS TI99/4A

Il movimento del TI BASIC

di Sergio Borsani

Il TI99/4A, come è noto, può essere programmato in TI BASIC, Extended BASIC, Assembler, Pascal e Logo che, verso la fine di quest'anno, uscirà nella versione italiana. Per i giochi o i programmi didattici che richiedono un intenso movimento il più adatto è l'Assembler che più si avvicina al linguaggio macchina ma che tuttavia è difficile da usare. I primi due linguaggi elencati all'inizio invece sono di gran lunga i più usati e si differenziano notevolmente per quanto riguarda la capacità di creare oggetti in movimento. L'Extended BASIC infatti è provvisto di sottoprogrammi, richiamabili con istruzioni CALL..., rivolti espressamente alla grafica e fornisce così la possibilità di creare un ambiente molto simile a quello dei più comuni videogames.

Mentre ho intenzione di illustrare questo argomento in un prossimo numero, desidero ora soffermarmi sul TI BASIC.

Questo linguaggio, residente nella console, è poco adatto alle animazioni grafiche e per utilizzarlo a questo scopo è necessario seguire alcune regole: (a), gli oggetti che si muovono devono essere piccoli, preferibilmente formati da un solo carattere; (b), il loro numero deve essere limitato; (c), i cicli che sono adibiti al movimento devono contenere solo le istruzioni essenziali, tutte le altre dovranno essere poste all'interno di essi.

Il movimento ovviamente si crea stampando uno space, CHR$ (32), nella vecchia posizione e collocando lo stesso oggetto in una posizione nuova. L'effetto è inferiore se si invertono le istruzioni cioè se prima si stampa il carattere nella nuova posizione e poi si cancella nella vecchia. Il movimento non può essere uniforme ma apparirà a scatti poichè l'oggetto balza da una riga o da una colonna ad un'altra. Per ottenere una maggiore velocità, i salti potranno comprendere più di una riga o di una colonna a scapito della già precaria uniformità del movimento.

Se l'oggetto è formato da più di un carattere esso si sposterà parte per parte e l'andatura assumerà un aspetto ondeggiante poichè il più delle volte alcuni caratteri che formano l'oggetto occuperanno già la nuova posizione mentre altri dovranno ancora essere trasferiti.

Anche se gli oggetti sono piccoli e formati da un solo carattere, ricordiamo che un programma TI BASIC può spostarli soltanto uno alla volta. È evidente che se si volesse creare un gioco come quello del calcio, dove è richiesto il movimento della palla e di molti giocatori, per spostare la linea d'attacco il programma dovrebbe compiere numerosi cicli per le proprie mosse e quelle dell'avversario rendendo l'esecuzione di una lentezza esasperante. Una complicazione deriva anche dal fatto che un carattere cancella un altro sul quale viene stampato, pertanto il ciclo che crea il movimento deve contenere le istruzioni perchè i caratteri non si sovrappongano o per ricrearli dopo un'eventuale sovrapposizione.

```
100 CALL CLEAR
110 CALL CHAR(128,"F")
120 CALL COLOR(13,2,2)
130 X1=16
140 Y1=12
150 CALL HCHAR(Y1,X1,128)
160 CALL KEY(0,K,S)
170 IF S=0 THEN 160
180 IF K=69 THEN 230
190 IF K=88 THEN 260
200 IF K=83 THEN 290
210 IF K=68 THEN 320
220 GOTO 160
230 Y2=Y1-1
240 X2=X1
250 IF Y2<1 THEN 160 ELSE 350
260 Y2=Y1+1
270 X2=X1
280 IF Y2>24 THEN 160 ELSE 350
290 X2=X1-1
300 Y2=Y1
310 IF X2<1 THEN 160 ELSE 350
320 X2=X1+1
330 Y2=Y1
340 IF X2>32 THEN 160
350 CALL HCHAR(Y1,X1,32)
360 CALL HCHAR(Y2,X2,128)
370 X1=X2
380 Y1=Y2
390 GOTO 160
```

Listato 1. *Un semplice esempio di animazione.*

Se si desidera controllare il movimento da tastiera bisogna inserire l'istruzione CALL KEY. Solitamente si associano i quattro tasti con le frecce ai movimenti nelle quattro direzioni; una tipica sequenza di istruzioni è quella del listato 1.

In questo esempio, premendo i quattro tasti con le frecce, un quadratino nero si sposta in tutte le direzioni.

È importante controllare il valore delle coordinate ed impedire la stampa quando il numero di riga è minore di 1 o maggiore di 24 e quando il numero di colonna è minore di 1 o maggiore di 32.

Il listato 2 è un programma didattico sulle frazioni. Si tratta di tagliare da un rettangolo una parte frazionaria controllando da tastiera un paio di forbici che appaiono sullo schermo. Il tasto (S) le sposta a sinistra, il tasto (D) a destra ed il tasto (T) provoca il movimento automatico che simula il taglio. Per ogni prova sono consentiti due tentativi, dopo due errori consecutivi il computer suggerisce la risposta esatta.

Il movimento del TI BASIC

Al termine dei quesiti viene indicato il livello raggiunto.

La tecnica di programmazione è conforme ai concetti su esposti. Una complicazione è dovuta al fatto che le forbici, formate da due o quattro caratteri, partono dall'esterno del rettangolo colorato, vi si addentrano lasciando alle spalle la traccia del taglio e poi escono dalla parte opposta passando per due volte dallo sfondo di un colore a quello di un altro.

Dopo aver utilizzato il programma, apparirà evidente come le animazioni grafiche siano importanti non solo per i giochi, ma trovino anche una loro precisa collocazione nei programmi educativi. ■

Listato 2. *Esempio di animazione applicato alla didattica.*

```
100 REM  PERSONAL SOFTWARE
110 REM  ******************
120 REM  *                *
130 REM  *    FRAZIONI    *
140 REM  *                *
150 REM  ******************
160 REM  di: Sergio Borsani
170 REM  versione: TI BASIC
180 CALL CLEAR
190 PRINT TAB(11);"FRAZIONI":::::::::
200 CALL CHAR(128,"0")
210 CALL CHAR(129,"0101010101010101")
220 A$="0010181800C0C0607"
230 B$="0301031E32263C18"
240 C$="0103030303030303"
250 D$="0301070D09090F06"
260 CALL CHAR(130,A$)
270 CALL CHAR(131,B$)
280 CALL CHAR(132,C$)
290 CALL CHAR(133,D$)
300 CALL CHAR(136,A$)
310 CALL CHAR(137,"0010303060600C0C0")
320 CALL CHAR(138,B$)
330 CALL CHAR(139,"800080F098C87830")
340 CALL CHAR(140,C$)
350 CALL CHAR(141,"0080808080808080")
360 CALL CHAR(142,D$)
370 CALL CHAR(143,"8000C0602020E0C0")
380 CALL CHAR(144,"00000000FF")
390 CALL CHAR(145,"FF8181")
400 CALL CHAR(146,"FF0101")
410 CALL COLOR(13,2,14)
420 CALL CLEAR
430 PRINT "HAI A DISPOSIZIONE UN PAIO"
440 PRINT "DI FORBICI PER TAGLIARE PAR-"
450 PRINT "TI FRAZIONARIE":::
460 PRINT "USA I SEGUENTI COMANDI:"::
470 PRINT "TASTO (D) PER SPOSTARE LE"
480 PRINT TAB(11);"FORBICI A DESTRA"
490 PRINT "TASTO (S) PER SPOSTARE LE"
500 PRINT TAB(11);"FORBICI A SINISTRA"
510 PRINT "TASTO (T) PER TAGLIARE.":::: 
520 PRINT TAB(8);"premi un tasto"
530 PRINT TAB(8);"per continuare"
540 CALL KEY(0,K,S)
550 IF S=0 THEN 540
560 PUNTI=0
570 NUME=1
580 DENO=2
590 GOSUB 1100
600 NUME=1
610 DENO=4
620 GOSUB 1100
630 NUME=1
640 DENO=3
650 GOSUB 1100
660 NUME=3
670 DENO=4
680 GOSUB 1100
690 NUME=2
```

Seguito listato 2.

```
700 DENO=3
710 GOSUB 1100
720 NUME=5
730 DENO=6
740 GOSUB 1100
750 NUME=5
760 DENO=12
770 GOSUB 1100
780 NUME=2
790 DENO=4
800 GOSUB 1100
810 NUME=3
820 DENO=6
830 GOSUB 1100
840 NUME=7
850 DENO=12
860 GOSUB 1100
870 CALL CLEAR
880 IF PUNTI<20 THEN 920
890 PRINT TAB(11);"OTTIMO!"::::::::
900 GOSUB 2730
910 GOTO 1030
920 IF PUNTI<15 THEN 960
930 PRINT TAB(12);"BENE!"::::::::
940 GOSUB 2730
950 GOTO 1030
960 IF PUNTI<10 THEN 1000
970 PRINT TAB(9);"SUFFICIENTE."::::::::
980 GOSUB 2500
990 GOTO 1030
1000 PRINT "TEMO CHE LA MATEMATICA NON"
1010 PRINT "SIA IL TUO FORTE."::::::::
1020 GOSUB 2630
1030 PRINT "VUOI CONTINUARE? (S/N)"
1040 CALL KEY(0,K,S)
1050 IF S=0 THEN 1040
1060 IF (K=83)+(K=115)THEN 560
1070 IF (K=78)+(K=110)THEN 2850
1080 GOTO 1040
1090 REM  *** ANIMAZIONE ***
1100 CALL CLEAR
1110 FOR J=3 TO 11
1120 CALL HCHAR(J,11,128,12)
1130 NEXT J
1140 F$="TAGLIA"
1150 FOR J=1 TO 6
1160 L=ASC(SEG$(F$,J,1))
1170 CALL HCHAR(17,10+J,L)
1180 NEXT J
1190 LN=LEN(STR$(NUME))
1200 LD=LEN(STR$(DENO))
1210 FOR J=1 TO LN
1220 L=ASC(SEG$(STR$(NUME),J,1))
1230 CALL HCHAR(16,19-LN+J,L)
1240 NEXT J
1250 CALL HCHAR(17,19,144)
1260 IF (LN=1)*(LD=1)THEN 1280
1270 CALL HCHAR(17,18,144)
1280 FOR J=1 TO LD
1290 L=ASC(SEG$(STR$(DENO),J,1))
1300 CALL HCHAR(18,19-LD+J,L)
1310 NEXT J
1320 XVECCHIO=11
1330 CALL HCHAR(13,XVECCHIO,136)
1340 CALL HCHAR(13,XVECCHIO+1,137)
1350 CALL HCHAR(14,XVECCHIO,138)
1360 CALL HCHAR(14,XVECCHIO+1,139)
1370 CALL KEY(0,K,S)
1380 IF S=0 THEN 1370
1390 IF (K=68)+(K=100)THEN 1440
1400 IF (K=83)+(K=115)THEN 1470
1410 IF (K=84)+(K=116)THEN 1600
1420 GOTO 1370
1430 REM  *** DESTRA ***
1440 XDIR=1
1450 GOTO 1480
1460 REM  *** SINISTRA ***
1470 XDIR=-1
1480 XN=XVECCHIO+XDIR
1490 IF (XN<11)+(XN>21)THEN 1370
1500 XNUOVO=XN
```

Seguito listato 2.

```
1510 CALL HCHAR(13,XVECCHIO,32,2)
1520 CALL HCHAR(14,XVECCHIO,32,2)
1530 CALL HCHAR(13,XNUOVO,136)
1540 CALL HCHAR(13,XNUOVO+1,137)
1550 CALL HCHAR(14,XNUOVO,138)
1560 CALL HCHAR(14,XNUOVO+1,139)
1570 XVECCHIO=XNUOVO
1580 GOTO 1370
1590 REM  *** TAGLIA ***
1600 X1=XVECCHIO
1610 X2=XVECCHIO+1
1620 CALL HCHAR(13,X1,32,2)
1630 CALL HCHAR(14,X1,32,2)
1640 CALL HCHAR(12,X1,136)
1650 CALL HCHAR(12,X2,137)
1660 CALL HCHAR(13,X1,138)
1670 CALL HCHAR(13,X2,139)
1680 CALL HCHAR(11,X1,130)
1690 CALL HCHAR(12,X1,138)
1700 CALL HCHAR(12,X2,139)
1710 CALL HCHAR(13,X1,32,2)
1720 CALL HCHAR(11,X1,132)
1730 CALL HCHAR(12,X1,142)
1740 CALL HCHAR(12,X2,143)
1750 CALL HCHAR(11,X1,130)
1760 CALL HCHAR(12,X1,138)
1770 CALL HCHAR(12,X2,139)
1780 CALL HCHAR(10,X1,130)
1790 CALL HCHAR(11,X1,131)
1800 CALL HCHAR(12,X1,32,2)
1810 FOR R=10 TO 4 STEP -1
1820 CALL HCHAR(R,X1,132)
1830 CALL HCHAR(R+1,X1,133)
1840 CALL HCHAR(R,X1,130)
1850 CALL HCHAR(R+1,X1,131)
1860 CALL HCHAR(R-1,X1,130)
1870 CALL HCHAR(R,X1,131)
1880 CALL HCHAR(R+1,X1,129)
1890 NEXT R
1900 CALL HCHAR(3,X1,132)
1910 CALL HCHAR(4,X1,133)
1920 CALL HCHAR(3,X1,130)
1930 CALL HCHAR(4,X1,131)
1940 CALL HCHAR(2,X1,136)
1950 CALL HCHAR(2,X2,137)
1960 CALL HCHAR(3,X1,131)
1970 CALL HCHAR(4,X1,129)
1980 CALL HCHAR(1,X1,136)
1990 CALL HCHAR(1,X2,137)
2000 CALL HCHAR(2,X1,138)
2010 CALL HCHAR(2,X2,139)
2020 CALL HCHAR(3,X1,129)
2030 FOR T=1 TO 100
2040 NEXT T
2050 COLO=10+12*NUME/DENO
2060 IF X1=COLO THEN 2380
2070 IF VOLTE=2 THEN 2170
2080 E$="ERRORE! RIFROVA"
2090 FOR J=1 TO 15
2100 L=ASC(SEG$(E$,J,1))
2110 CALL HCHAR(21,8+J,L)
2120 NEXT J
2130 FOR T=1 TO 1000
2140 NEXT T
2150 VOLTE=2
2160 GOTO 1100
2170 VOLTE=0
2180 EE$="ERRORE!"
2190 FOR J=1 TO 7
2200 L=ASC(SEG$(EE$,J,1))
2210 CALL HCHAR(21,10+J,L)
2220 NEXT J
2230 GOSUB 2630
2240 RR$="GUARDA LA RISPOSTA ESATTA."
2250 FOR J=1 TO 26
2260 L=ASC(SEG$(RR$,J,1))
2270 CALL HCHAR(23,2+J,L)
2280 NEXT J
2290 CALL HCHAR(12,11,145)
2300 CALL HCHAR(12,12,146,11)
2310 CALL HCHAR(13,COLO,136)
```

Seguito listato 2.

```
2320 CALL HCHAR(13,COLO+1,137)
2330 CALL HCHAR(14,COLO,138)
2340 CALL HCHAR(14,COLO+1,139)
2350 FOR T=1 TO 3000
2360 NEXT T
2370 GOTO 2490
2380 ES$="ESATTO!"
2390 FOR J=1 TO 7
2400 L=ASC(SEG$(ES$,J,1))
2410 CALL HCHAR(21,10+J,L)
2420 NEXT J
2430 GOSUB 2500
2440 IF VOLTE=2 THEN 2470
2450 PUNTI=PUNTI+2
2460 GOTO 2480
2470 PUNTI=PUNTI+1
2480 VOLTE=0
2490 RETURN
2500 CALL SOUND(400,440,8)
2510 CALL SOUND(200,392,8)
2520 CALL SOUND(200,440,7)
2530 CALL SOUND(400,587,6)
2540 CALL SOUND(200,523,5)
2550 CALL SOUND(200,587,4)
2560 CALL SOUND(400,494,3)
2570 CALL SOUND(200,440,4)
2580 CALL SOUND(200,494,5)
2590 CALL SOUND(500,392,6)
2600 FOR T=1 TO 300
2610 NEXT T
2620 RETURN
2630 CALL SOUND(400,440,6)
2640 CALL SOUND(150,440,6)
2650 CALL SOUND(400,349,6)
2660 CALL SOUND(150,330,6)
2670 CALL SOUND(400,175,6)
2680 CALL SOUND(150,165,6)
2690 CALL SOUND(1000,131,6)
2700 FOR T=1 TO 800
2710 NEXT T
2720 RETURN
2730 CALL SOUND(300,698,2)
2740 CALL SOUND(300,932,2)
2750 CALL SOUND(300,784,2)
2760 CALL SOUND(300,880,2)
2770 CALL SOUND(150,932,2)
2780 CALL SOUND(150,880,2)
2790 CALL SOUND(150,784,2)
2800 CALL SOUND(150,880,2)
2810 CALL SOUND(500,698,2)
2820 FOR T=1 TO 300
2830 NEXT T
2840 RETURN
2850 CALL CLEAR
2860 END
```

DEBUG

Dal listato BASIC del Programmer's tool kit, pubblicato nel n. 12/13 è saltata una linea fondamentale per il corretto funzionamento del programma. Occorre aggiungere la seguente istruzione:
1305 DATA A9, 2C, 20, D2, FF, A5, FE.

PERSONAL SOFTWARE

DEBUG

Wei-ch'i

Come i nostri lettori avranno avuto modo di notare, sullo scorso numero di **Personal Software**, in luogo del listato del programma che vi veniva descritto, è apparso il listato di un programma di statistica ad una dimensione. Scusandoci con i lettori di **Personal Software**, pubblichiamo su questo numero il listato scomparso. ■

```
0 CLR:PRINTCHR$(142):DIMW$(25):W$="[illegible]":[illegible]:FB$="[illegible]"
3 FN$=FB$:BH$="00§ ■00§ ■00":NH$=BH$:MO$="0000":MO=0:PE=0:PN=0:BU=10
4 NH=0:BH=0:T$(1)="COLONNA:  ■■":T$(2)="RIGA    :  ■■":F=0
5 FORI=1TO25:W$(I)=LEFT$(W$,I):NEXT
7 PRINT"■"W$(5)TAB(3)"AVETE ALTRE PARTITE IN CORSO":INPUTR$
8 IFLEFT$(R$,1)="S"THENFF=1:GOSUB7000:GOSUB7025:TI$=T$:GOTO130
10 PRINT"■"W$(5)TAB(3)"DIMENSIONI DELLA SCACCHIERA 5-23"
20 PRINT"■"TAB(3)"(INTERSEZIONI PER LATO):";:INPUTL:IFL<5ORL>23THEN10
25 DIMS(L,L),X(BU),Y(BU),M(BU):FORI=1TOL:FORJ=1TOL:S(I,J)=0:NEXTJ,I
30 B=INT(L/2):H=INT(L/2-B):B0=B:H0=H:PRINTTAB(3)"■HANDICAP (S/N)";:INPUTR$
40 IFLEFT$(R$,1)="N"THEN50
45 GOSUB2500:PRINTTAB(5)"■ 2 - "NH")";
47 GH=1:INPUTH:IFH<2ORH>NHTHEN45
50 GOSUB550
94 IFLEFT$(R$,1)="N"THEN130
100 GOSUB6000:FORK=1TOH:H=H-1:PRINTW$(23)TAB(L6)"■HANDICAP■"
110 FG=0:GOSUB2000:GOSUB3000:S(X,Y)=M:GOSUB6000:IFFG=1THENFG=0:GOTO110
115 NEXT:GH=0:M=1
130 PRINTW$(2)TAB(L0+17)"BIANCO"SPC(1)PB$:PRINTW$(6)TAB(L0+18)BH$
140 PRINTW$(8)TAB(L0+17)"MOSSE"SPC(1)MO$:PRINTW$(10)TAB(L0+18)IT$
145 PRINTW$(12)TAB(L0+17)" NERO "SPC(1)PN$:PRINTW$(16)TAB(L0+18)NH$
150 IFFF=0THENTI$="000000":GOSUB9910
160 IFN=0ORB=0THENGOSUB8000:END
170 FG=0:GOSUB2000:GOSUB9000:IFFL=2THEN160
175 GOSUB3000:GOSUB6000:MO=MO+1:S(X,Y)=M:IFFG=1THENFG=0:GOTO170
180 GOSUB9000:IFM=-1THENN=N-1
190 GOSUB9000:IFM=1THENB=B-1
197 IFFF<>1THENGOSUB13000
198 IFFF=1THENFF=0
199 GOSUB17000:GOSUB20000
200 GOSUB9910:GOSUB9000:M=-M:GOTO160
550 PRINT"■":FORI=1TOL:POKE32768+I,I:NEXT:PRINTW$(2)CHR$(65)",":FORI=1TOL-2
560 POKE32808+I,114:NEXT:PRINTW$(2)TAB(L)",";CHR$(65):FORJ=1TOL-2
570 PRINTW$(J+2)CHR$(65+J)"[illegible]"TAB(L)"[illegible]"CHR$(65+J):NEXT:PRINTW$(1+L)CHR$(64+L)"[illegible]
580 FORI=1TOL-2:POKE32769+L*40+I,113:NEXT:PRINTW$(1+L)TAB(L)"[illegible]"CHR$(64+L)
590 FORI=1TOL-2:FORJ=1TOL-2:POKE32769+(J+1)*40+I,91:NEXTJ,I
591 FORI=1TOL:POKE32768+(L+1)*40+I,I:NEXT:GOSUB1000:L0=INT(L/2):L6=L0+16
592 FORI=0TO16:FORJ=L6TOL0+27:POKE32768+I*40+J,160:NEXTJ,I:RETURN
1000 IF(L/2)*2=INT(L/2)*2THEN1500
1010 NH=9:K=INT(L/2)-3:ONKGOTO1011,1012,1013,1014,1015,1016,1017,1018
1011 SP=2:B0=1:GOTO1020
1012 SP=3:B0=1:GOTO1020
1013 SP=3:B0=2:GOTO1020
1014 SP=4:B0=2:GOTO1020
1015 SP=4:B0=3:GOTO1020
1016 SP=5:B0=3:GOTO1020
1017 SP=5:B0=4:GOTO1020
1018 SP=6:B0=4
1020 FORI=1TO3:PRINTW$(B0+2+(I-1)*(SP+1));:FORJ=1TO3
1030 PRINTTAB(B0+1+(J-1)*(SP+1))"■ ■";:NEXTJ,I:RETURN
1500 NH=4:K=(L/2)-3:ONKGOTO1501,1502,1503,1504,1505,1506,1507,1508
1501 SP=2:B0=2:GOTO1510
1502 SP=4:B0=2:GOTO1510
1503 SP=4:B0=3:GOTO1510
1504 SP=6:B0=3:GOTO1510
1505 SP=6:B0=4:GOTO1510
1506 SP=8:B0=4:GOTO1510
1507 SP=8:B0=5:GOTO1510
1508 SP=10:B0=5
1510 FORI=1TO2:PRINTW$(B0+2+(I-1)*(SP+1));:FORJ=1TO2
1520 PRINTTAB(B0+1+(J-1)*(SP+1))"■ ■";:NEXTJ,I:RETURN
2000 FORF=1TO2:O$(F)=T$(F):NEXT:IFM=1THEN:FORF=1TO2:O$(F)="■"+T$(F)+"■":NEXT
2005 FL=0
2010 PRINTW$(19)TAB(L6)O$(1):N=19:TB=L6+9:GOSUB2100
2015 IFFL=1THENFL=0:GOTO2000
2020 IFFL=2THENRETURN
2025 IFASC(R$)<65ORASC(R$)>64+LTHENGOSUB9000:GOTO2010
2027 Y=ASC(R$)-64
2030 PRINTW$(21)TAB(L6)O$(2):N=21:TB=L6+9:GOSUB2100
2035 IFFL=1THENFL=0:GOTO2000
2037 IFFL=2THENRETURN
2039 IFASC(R$)<65ORASC(R$)>64+LTHENGOSUB9000:GOTO2030
2040 X=ASC(R$)-64:RETURN
2100 GETR$:IFR$=""THENGOSUB9000:GOTO2100
2110 PRINTW$(N)TAB(TB)R$:IFR$="@"THENTT$=TI$:GOSUB4000:END
2120 IFR$="↑"THENGOSUB5000:FL=1:RETURN
2125 IFR$="←"THENGOSUB15000:FL=2
2130 RETURN
3000 IFS(X,Y)<>0THENPRINTW$(24)TAB(L6)"ERRORE":FG=1:RETURN
3010 GOSUB6000:CR=87:IFM=1THENCR=81
3020 IFPEEK(32768+X*40+Y)=160ANDCR=87THENCR=209:GOTO3040
3030 IFPEEK(32768+X*40+Y)=160ANDCR=81THENCR=215
3040 FORD=1TO15:POKE32768+X*40+Y,160
3050 POKE32768+X*40+Y,32:POKE32768+X*40+Y,CR:NEXT:RETURN
4000 PRINT"■PREPARATE UNA CASSETTA E RIAVVOLGETE-":PRINT"LA.":
4005 INPUT" PRONTI";R$:IFLEFT$(R$,1)<>"S"THEN4000
4010 OPEN1,1,2,"WEI-CH'I. FILE DATI"
4015 PRINT#1,M:PRINT#1,L:PRINT#1,L0:PRINT#1,N:PRINT#1,L6:PRINT#1,H0:PRINT#1,B0
4020 PRINT#1,B:PRINT#1,NH:PRINT#1,BH:PRINT#1,PB:PRINT#1,PN:PRINT#1,MO
4030 TI$=TT$:PRINT#1,TI$:PRINT#1,BH:PRINT#1,NH
4035 PRINT#1,BU:FORPK=1TOBU:PRINT#1,X(PK):PRINT#1,Y(PK):PRINT#1,M(PK):NEXT
4040 PRINT#1,OM:FORI=1TOL:FORJ=1TOL:PRINT#1,S(I,J):NEXTJ,I:CLOSE1:END
5000 [illegible]:PRINTW$(13)TAB(L6)"[illegible]"
5005 PRINTTAB(L6)"[illegible]"
5010 PRINTTAB(L6);:INPUTE:GOSUB6000:[illegible]TOPE
5020 [illegible]:PRINTW$(24)TAB(L6)"ERRORE":GOSUB6000:GOTO5020
5030 GOSUB6000:GOSUB11000:GOSUB12000:S(X,Y)=0:NEXT
5040 [illegible]
6000 FORI=18TO24:GOSUB9000:FORJ=L6TOL0+27:POKE32768+I*40+J,32:NEXTJ,I:RETURN
6500 NH=9:IF(L/2)*2=INT(L/2)*2THENNH=4
6510 RETURN
7000 PRINT"■RIAVVOLGETE LA CASSETTA CONTENENTE IL":PRINT"FILE DATI.":
7005 INPUT" PRONTI";R$:IFLEFT$(R$,1)<>"S"THEN7000
7010 OPEN1,1,0,"WEI-CH'I. FILE DATI":INPUT#1,M:INPUT#1,L:INPUT#1,L0:INPUT#1,N
7015 INPUT#1,L6:INPUT#1,H0:INPUT#1,B0
7020 INPUT#1,B:INPUT#1,NH:INPUT#1,BH:INPUT#1,PB:INPUT#1,PN:INPUT#1,MO
7030 INPUT#1,TI$:TT$=TI$:INPUT#1,BH:INPUT#1,NH:INPUT#1,BU
7035 DIMS(L,L),X(BU),Y(BU),M(BU)
7037 FORPK=1TOBU:INPUT#1,X(PK):INPUT#1,Y(PK):INPUT#1,M(PK):NEXT
7040 INPUT#1,OM:FORI=1TOL:FORJ=1TOL:INPUT#1,S(I,J):NEXTJ,I:CLOSE1
7050 PB$=RIGHT$(STR$(PB),LEN(STR$(PB))-1):PB$=RIGHT$("000"+PB$,3)
7060 PN$=RIGHT$(STR$(PN),LEN(STR$(PN))-1):PN$=RIGHT$("000"+PN$,3)
7070 MO$=RIGHT$(STR$(MO),LEN(STR$(MO))-1):MO$=RIGHT$("0000"+MO$,4)
7080 IT$=LEFT$(TI$,2)+"■ ■"+MID$(TI$,3,2)+"■ ■"+RIGHT$(TI$,2):RETURN
7085 GOSUB550
7090 FORX=1TOL:FORY=1TOL:IFS(X,Y)=1THENCR=81
7100 IFS(X,Y)=-1THENCR=87
7110 IFS(X,Y)=0THEN7130
7120 GOSUB3020
7130 NEXTY,X:GOSUB9910:RETURN
8000 FORI=1TO50000:END
9000 H(1)=VAL(RIGHT$(TI$,2)):H(2)=VAL(MID$(TI$,3,2)):H(3)=VAL(LEFT$(TI$,2))
9005 IFGH=1THENRETURN
9010 DV=H(1)+H(2)*60+H(3)*3600:IFM=1THEN9500
9020 NH=DV-BH:NH(1)=INT(NH/3600):NH(2)=INT(NH/60-NH(1)*60)
9030 NH(3)=INT(NH-NH(1)*3600-NH(2)*60):FORHN=1TO3
9035 NH$(HN)=RIGHT$(STR$(NH(HN)),LEN(STR$(NH(HN)))-1)
9037 NH$(HN)=RIGHT$("00"+NH$(HN),2):NEXT
9040 NH$=NH$(1)+"■ ■"+NH$(2)+"■ ■"+NH$(3):PRINTW$(16)TAB(L0+18)NH$
9050 GOTO9530
9500 BH=DV-NH:BH(1)=INT(BH/3600):BH(2)=INT(BH/60-BH(1)*60)
9510 BH(3)=INT(BH-BH(1)*3600-BH(2)*60):FORHN=1TO3
9515 BH$(HN)=RIGHT$(STR$(BH(HN)),LEN(STR$(BH(HN)))-1)
9517 BH$(HN)=RIGHT$("00"+BH$(HN),2):NEXT
9520 BH$=BH$(1)+"■ ■"+BH$(2)+"■ ■"+BH$(3):PRINTW$(6)TAB(L0+18)BH$
9530 GOSUB7080:PRINTW$(10)TAB(L0+18)IT$:RETURN
9910 MO$=RIGHT$(STR$(MO),LEN(STR$(MO))-1):MO$=RIGHT$("0000"+MO$,4)
9911 N0$=RIGHT$(STR$(N),LEN(STR$(N))-1):N0$=RIGHT$("0000"+N0$,3):N9=N0-N
9912 B0$=RIGHT$(STR$(B),LEN(STR$(B))-1):B0$=RIGHT$("0000"+B0$,3):B9=B0-B
9913 N9$=RIGHT$(STR$(N9),LEN(STR$(N9))-1):N9$=RIGHT$("0000"+N9$,3)
9914 B9$=RIGHT$(STR$(B9),LEN(STR$(B9))-1):B9$=RIGHT$("0000"+B9$,3)
9915 PRINTW$(4)TAB(L0+18)B0$TAB(L0+23)B9$
9916 PRINTW$(8)TAB(L0+23)MO$
9917 PRINTW$(14)TAB(L0+18)N0$TAB(L0+23)N9$
9920 RETURN
10000 FORI=1TOL:FORJ=1TOL:IFS(I,J)=0THEN10020
10010 X=I:Y=J:M=S(I,J):GOSUB3010
10020 NEXTJ,I:RETURN
11000 CR=91:IFX=1THENCR=114
11010 IFX=LTHENCR=113
11020 IFY=1THENCR=107
11030 IFY=LTHENCR=115
11040 IFX=1ANDY=1THENCR=112:GOTO11075
11050 IFX=1ANDY=LTHENCR=110:GOTO11075
11060 IFX=LANDY=1THENCR=109:GOTO11075
11070 IFX=LANDY=LTHENCR=125:GOTO11075
11073 KP=PEEK(32768+X*40+Y):IFCR=91AND(KP=215ORKP=209ORKP=160)THENCR=160
11075 IFX=0ANDY=0THENRETURN
11076 FORO=1TO15:POKE32768+X*40+Y,160:POKE32768+X*40+Y,32
11080 POKE32768+X*40+Y,CR:NEXT:RETURN
12000 IFS(X,Y)=0THENRETURN
12005 IFM=-1THEN12500
12010 PN=PN+1:PN$=RIGHT$(STR$(PN),LEN(STR$(PN))-1):PN$=RIGHT$("000"+PN$,3)
12020 PRINTW$(12)TAB(L0+24)PN$:RETURN
12500 PB=PB+1:PB$=RIGHT$(STR$(PB),LEN(STR$(PB))-1):PB$=RIGHT$("000"+PB$,3)
12510 PRINTW$(2)TAB(L0+24)PB$:RETURN
13000 OM=MO:IFMO>BUTHENGOSUB14000:OM=BU
13005 IFOM=0THENRETURN
13010 X(OM)=X:Y(OM)=Y:M(OM)=M:RETURN
14000 FORPK=1TOBU-1:X(PK)=X(PK+1):Y(PK)=Y(PK+1):NEXT:RETURN
15000 GOSUB6000:PRINTW$(19)TAB(L6)"INDIETRO DI"
15005 PRINTTAB(L6)"QUANTE MOS-"
15010 PRINTTAB(L6)"SE";:INPUTMB:GOSUB6000:IFMB>=OMTHEN15000
15015 FORPK=0TOMB-1:X=X(OM-PK):Y=Y(OM-PK):MO=MO-1:M=M(OM-PK)
15018 GOSUB16000:GOSUB9910:S(X,Y)=0:GOSUB11000:NEXT:OM=OM-MB
15020 GOSUB6000:RETURN
16000 IFM=-1THEN16500
16010 B=B+1:GOSUB9910:RETURN
16500 N=N+1:GOSUB9910:RETURN
17000 FORDX=-1TO1:X0=X+DX:IFX0>LORX0<0THEN17500
17003 FORDY=-1TO1:Y0=Y+DY:IFY0>LORY0<0THEN17300
17005 IF(DX=DYORDX=-DY)THEN17300
17010 GOSUB19000
17300 NEXTDY
17500 NEXTDX:RETURN
17800 GOSUB6000:GOSUB11000:GOSUB12000:S(X,Y)=0:RETURN
17900 X5=X:Y5=Y:M5=M:X=X0:Y=Y0:M=S(X0,Y0):GOSUB17800:X=X5:Y=Y5:M=M5:RETURN
18000 IFX0=1ORX0=LTHENTC=TC-1
18010 IFY0=1ORY0=LTHENTC=TC-1
18020 RETURN
19000 CT=0:TC=4:GOSUB18000:FORXD=-1TO1:X9=X0+XD
19010 FORYD=-1TO1:IFXD=0ANDYD=0THEN19100
19020 Y9=Y0+YD:IF(XD=YDORXD=-YD)THEN19100
19030 IFX9>LORX9<0ORY9>LORY9<0THEN19100
19035 IFS(X0,Y0)=0THEN19100
19040 IFS(X9,Y9)=-S(X0,Y0)THENCT=CT+1
19100 NEXTYD
19105 NEXTXD
19110 IFCT=TCTHENGOSUB17900
19120 RETURN
20000 X0=X:Y0=Y:GOSUB19000:RETURN
50000 FORI=1TOL:FORJ=1TOL:PRINTS(I,J);:NEXT:PRINT:NEXT
```

Servizio programmi

Per alcuni dei programmi pubblicati, *Personal Software* mette a disposizione dischi e nastri già registrati, realizzati in collaborazione con l'autore. Potete ottenerli in contrassegno, pagando direttamente al postino la cifra indicata, spedendo il tagliando pubblicato in fondo alla pagina.

N.	Sistema	Programmi	Supporto	pubblicato in *Personal Software* n.	Prezzo
1	Apple II+	La carta del cielo Collisione	floppy 5" DOS 3.3	3 pag. 83 3 pag. 93	30.000
2	TRS-80 mod. I	Backgammon	floppy 5" DOS 2.3	3 pag. 89	25.000
3	PET/CBM 3032/4032	Editor/Assembler in Basic	floppy 5" 3032/4032+3040/4040	2 pag. 33	40.000
4	Apple II+	Interi in precisione multipla Grafica 3D	floppy 5" DOS 3.3	4 pag. 17 4 pag. 47	40.000
5	PET/CBM 3032/4032	Gioco del calcio	floppy 5" 3032/4032+3040/4040	4 pag. 67	25.000
6	Apple II +	Pretty Printer Shape Table	floppy 5" DOS 3.3	5 pag. 27 5 pag. 58	30.000
7	Apple II +	Data base modulare	floppy 5" DOS 3.3	7 pag. 47	25.000
8	PET/CBM 3032/4032	Wei-ch'i	cassetta	12/13 pag. 34	20.000
9	C 64	Tool-kit	cassetta	1 pag. 18	35.000

Spedire in busta chiusa a

PERSONAL SOFTWARE
Servizio Programmi
Via Rosellini 12
20124 Milano

Inviatemi i seguenti dischi di *Personal Software*

n. ______________________

per un totale di lire ______________ + L. 2.000 come contributo fisso spese di spedizione che pagherò al postino alla consegna del pacco.

____
Cognome e nome

____
Indirizzo

____
Cap., Località

____
Firma

PICCOLI ANNUNCI

Apple

Vendo Apple II 48K L. 1.200.000 Disk Drive con controller e Drive aggiuntivo. Disponibili stampante Epson MX 100 e numerose interfacce. Programmi in omaggio agli acquirenti.
Orlando Sergio - Via Dei Ciclamini, 37 - 20100 Milano - Tel. 02/4151963.

Vendo Lemon II 48K, monitor 12", disk drive per passaggio sistema superiore, 4 mesi vita L. 2.800.000 trattabili.
Nervi Roberto - Via Famagosta, 26/4 - 17100 Savona - Tel. 019/32753.

Vendo per Apple II "Supertoto 1.0", superprogramma totocalcio inedito, 3 diverse opzioni di selezioni incrociate; elaborazione superveloce delle colonne utili, output su video o stampante. L. 70.000.
Rossi Roberto - Via Lario, 26 - 20159 Milano - Tel. 6070236.

Vendo stampante termica silentype per Apple II. Vera occasione! L. 500.000 compresa interfaccia. Telefonare ore pasti.
Bova Domenico - Via Bergognone, 31 - 20144 Milano - Tel. 8394107.

*Scambio/*vendo bellissimi giochi (spaziali e di intelligenza) per Apple II, per chiarimenti ecco alcuni giochi: stellar invaders, Sargon II, Swasmbuckler, Paddle Fun, Santa Paravia, Decathlon, ecc.
Serenari Gabriele - Via Ca' Pellicelli, 1 - 42020 Albinea (RE) - Tel. 0522/599094.

Vendo The last one versione Apple II completo di manuali in italiano a L. 300.000.
Telefonare allo 0583/584795 ore pasti oppure allo 0583/331528 ore ufficio.
Betti Marco - Via Villa Altieri - 55100 Lucca - Tel. 584795.

Apple vendo con TV Sony Color, 2 Drivers, stampante Epson 100, Langvage Card, programmi più importanti completi di documentazione e diverse Card per applicazioni particolari. Telefonare 12-14.
Dimant Fabio - Via Raibolini, 33/7 - 40069 Zola Predosa (BO) - Tel. 051/273277.

Commodore

Vendo per 2001, 3032, 4032, programmi bioritmo Othello, Dama, Poker, Blackjack su cassetta L. 25.000 disco L. 30.000, telefonare ore serali.
Marchesotti Piero - Via M. Sabotino, 20 - 15067 Novi Ligure (AL) - Tel. 0143/75608.

Compro per VIC 20 configurazione base, listati o cassette per giochi e/o altro a prezzo interessante. Inviatemi le vostre liste, risponderò a tutti.
Li Bassi Lorenzo - Via S. Gerardo dei Tintori, 19 - 20046 Biassono (MI).

Vendo programmi per Commodore 64 giochi fotocopie listati inviatemi busta affrancata con vostro indirizzo vi spedirò elenco dettagliato L. 2.000 listato + 500 spese postali anche cassette.
Taccucci Claudio - Via dell'Acquedotto Paolo, 163 - 00168 Roma - Tel. 06/6795816.

Vendo a L. 30.000 (prezzo listino L. 85.000) programma "Frogger" per CBM 64 o cambio con altri programmi su cassetta. Cerco inoltre floppy disk VC 1541 solo se a un prezzo veramente favorevole.
Ferlin Federico - Via Fasolo, 28 - 35036 Montegrotto Terme (PD) - Tel. 794400.

Vendo per VIC 20 non espanso, cassetta di programmi musicali, contenente 10 brani celebri, eseguiti a 2 o 3 voci, tutti di ottimo effetto, a sole L. 20.000, spese e cassetta comprese.
Livi Roberto - Via Bissolati, 5 - 61100 Pesaro - Tel. 0721/65873.

Vendo/cambio programma magazzino VIC fino 500 articoli su disco con spiegazioni in italiano altro gestione condominio 16 K su disco 1-4 scale 400 condomini.
Locatelli Elio - Via N. Ardoino, 9/41 - 16151 Genova - Tel. 010/421701 (serali).

Cambio software su cassetta per PET 2001 New ROM. Inviatemi lista invierò la mia. Cerco programmi RTTY. Cerco schema PET 2001 e notizie su espansione di memoria sul C.S. interno. Telefonare 19,30-20.
Zafferani Vincenzo - Via Rancaglia - 47031 Serravalle Rep. S. Marino - Tel. 0541/901296.

Cerco 16K per VIC 20 a buon prezzo. Cerco stampante in ottimo stato.
Vimercati Mario - Via Varesina, 54 - 22079 Villaguardia (CO) - Tel. 031/480630.

Vendo per Commodore 64 "Screen Graphics 64". Tra le altre cose, aggiunge 24 comandi al Basic residente. Telefonatemi allo 039/734144 ore pasti. Franco (comprensivo di manuale e demo).

Compro VIC 20 + registratore + 16 K RAM + 3 K superexpander + manuali in italiano e cavi di connessione prezzo max. L. 600.000 - ore ufficio.
Vaccaielli Franco - Via Macerata, 71 - 00176 Roma - Tel. 270977.

Regalo registratore C2N + espansione 8 kbyte + vari programmi su cassette a chi acquista il mio VIC 20 (8 mesi di vita) al prezzo di vendita attuale (IVA inclusa).
Telefonare ore serali 004191/713167.
Russo Paolo - Via Statale, 36 - 21030 Marchirolo (VA) - Tel. 0332/722949.

CBM 64: cerco software di termotecnica, "373", impianti riscaldamento condizionamento, ventilazione, energia solare.
Cerco inoltre gestione magazzino.
Fornaciari Dino - Villaggio Dante, 30 - 52100 Arezzo - Tel. 0575/351451 (ore pasti).

Vendo VIC 20 + registratore Commodore + 8 K + 3 K + 3 K graphic + Turtle Graphics + Jelly Monster + Poker + Radar Ratrace + Slot + Jupiter Lander + manuali in italiano e tantissime riviste con programmi a L. 800.000 trattabili.
Rudella Bruno - Via Verdi, 31 - 24040 Arcene (BG) - Tel. 035/878594.

Cambio per VIC 20 programmi in LM dispongo di molti programmi (circa 80) anche inediti come sidewinder, swarm, VE120, Clitters, Cyclons e anche un monitor in LM completo di tutte le funzioni (+8 K).
Smilovich Maurizio - Via Del Carpineto, 16/1 - 34148 Trieste - Tel. 34148.

Vendo cassette per il VIC 20 con vari programmi incisi a modestissimi prezzi. I programmi sono scritti in linguaggio macchina.
Roberto Silva - Via Cagnola, 3 - 20154 Milano - Tel. 317228 (ore serali).

Vendo VIC 20 + super expander grafica 3K RAM, in perfette condizioni, in garanzia, usato poco, istruzioni in italiano, per passaggio a sistema superiore, a L. 380.000.
Telefonare dalle ore 19.000 in poi.
Proserpio Giancarlo - Via Besana, 6 - 20036 Meda (MI) - Tel. 0362/229750.

Vendo VIC 20 Commodore in perfette condizioni, usato pochissimo, completo di tutti gli accessori a L. 340.000.
Sangirardi Fabio - Via G. Vitelleschi, 6 - 00193 Roma - Tel. 06/6541818.

Per CBM 64 vendo programma di Word Processing con potente editor in LM, un versatile data base e un programma per la gestione delle spese familiari a L. 10.000 ciascuno. Richiesta stampante.
Monti Luca - Via Postcastello, 8 - 21013 Gallarate (VA) - Tel. 0331/792755.

Vendo e scambio fantastici programmi per VIC 20 cerco espansioni 8K, 16K massima serietà.
Ravagni Giulio - Via C.so Rosmini, 63 - 38068 Rovereto (TN) - Tel. 0464/34475.

Vendo VIC 20 + interfaccia registratore + cavi + manuali al migliore offerente. Tutto in condizioni veramente perfette.
Schianchi Massimo - Via G. Miranda, 3 - 80131 Napoli - Tel. 081/463025.

Compro cambio vendo programmi ad alta risoluzione grafica per Commodore 64.
Bacchetta Guglielmo - Casella Postale, 253 - 60035 Jesi (AN) - Tel. 0731/56705.

VIC 20 + registratore 2CN + espansione memoria e grafica alta risoluzione + manuali + cavetti + tutto il software fatto su cassette (giochi - archivi ecc.) a L. 500.000 telefonare ore pasti.
Albanesi Stefano - Via Leopardi, 10 - 20123 Milano - Tel. 8055804.

Vendo per VIC 20 3 cartucce (Avvenger, Starbattol, Radar) per L. 60.000 e Super Expander + programmi a L. 60.000, usati 2 mesi.
Maiocchi Fausto - Via Papa G. XXIII, 8 - 20082 Binasco (MI) - Tel. 9054059.

Scambio/vendo programmi per VIC 20, rispondo a tutti, mandatemi le vostre liste, vi invierò le mie, scrivere o telefonare (dalle 7 di sera).
Bottoni Gianfranco - Via Uglietti, 16 - 28067 Pernate (NO) - Tel. 0321/436102.

Cerco programmi di qualsiasi genere per VIC 20 a bassissimo costo da persone che se ne vogliono disfare o altri scrivere a:
Burattelli Gianluca - Via G. di Vittorio, 2 - 58022 Follonica (GR) - Tel. 0566/42445.

PICCOLI ANNUNCI

Vendo/cambio programmi 8 K su cassetta per VIC 20: Forth (+ manuale), VIC Graf, Text Editor, Avenger, Poker, Jupiter Lander, Frog, Sargon II Chess, Bonzo, Boss, Sub Chase, Galaxian, Defender, Rat Race, Alien.
Ariatti Claudio - Via Domodossola, 6 - 40139 Bologna - Tel. 051/544699.

Vendo programmi per VIC 20 a 3, 6, 8 o 16 K (giochi, utility, grafici) a prezzi da concordare, tramite invio cassetta o listato, per lista inviare L. 1.000. Massima serietà assicuro risposta a tutti.
Gollo Andrea - C.so Correnti, 37 - 10136 Torino.

Vendo per PET-CBM 3032 programmi di ingegneria: Stress 373 capurso, stampa Kani Word III Sisma e altri a prezzi veramente eccezionali.
Leonelli Paolo - Via A. Manzoni, 108 (presso Liverani) - 48010 Barbiano (RA) - Tel. 0545/78007.

Vendo, cambio, compro bellissimi programmi per CBM 64: Utility, Arcade Games, ecc. ecc. Telefonate!!!
Masini Patrizio - Via del Castagnone, 68 - 15048 Valenza Po (AL) - Tel. 0131/953695.

Vendo per CBM 64 software scientifico, Utility, giochi e vari; con possibilità di personalizzare i programmi a richiesta si invia gratuitamente il catalogo, tutti i programmi sono su cassetta.
Antoniucci Luciano - Via C. Goldoni, 7 - 05100 Terni - Tel. 0744/421274.

Sinclair

Causa passaggio sistema superiore vendo ZX81 espanso a 32 K RAM, con alimentatore potenziato e stabilizzato per evitare riscaldamenti del computer, a scelta collegabile a TV o a monitor in inverse-video.
Neri Stefano - Via S. Mario della Grotticella, 4/C - 01100 Viterbo - Tel. 0761/32442.

Vendo software per ZX81 con e senza espansione su listato. Scrivetemi per ricevere elenco dei programmi. Rispondo a tutti.
Vanoletti Paolo - Via Pindemonte, 1 - 20100 Milano.

Sinclair ZX Spectrum disponendo di un notevole archivio software vendo a L. 10.000 cassette registrate con 5 giochi a scelta Tel. 045/568649 ore pasti.
Parbuono Ivano - Via A. di Cambio - 37138 Verona.

Vendo ZX Spectrum 48 K usato pochissimo L. 420.000. Telefonare ore serali.
Spero Marcello - Via S. Polo, 2542 - 30125 Venezia - Tel. 041/37085.

Vendo Sinclair ZX 81 perfetto, garanzia in bianco e imballaggio con tutti gli accessori, manuali inglese e italiano, alimentatore maggiorato 1,2 ampere, espansione 16K + libro 66 programmi, cassetta programmi vari tutto a L. 220.000. Qualsiasi prova. Telefonare ore 20-21.
Piu Maurizio - Via M. Fanti, 21/51 - 16149 Genova-Sampierdarena - Tel. 010/418503.

Vendo ZX81 + alimentatore + cavi + manuale a L. 100.000 ottime condizioni usato pochissimo causa passaggio a sistema superiore.
Valle Paolo - V.le Matteotti, 73 - 27100 Pavia - Tel. 0382/32264.

Vendo ZX81, esp. 32K, mother board con 4 connettori, sound board, con ampli B.F., tastiera premente a reed il tutto in un contenitore d'alluminio, 2 manuali e programmi prezzo da stabilire.
Alziati Saverio - Via Dei Platani, 16/34 - 20020 Arese (MI) - Tel. 02/9383169.

Compro ZX Printer usata a prezzo modico.
Faraboschi Paolo - Via Boccadasse, 16/17 - 16146 Genova - Tel. 010/306806 (ore pasti).

Vendo programmi per ZX Spectrum 16/48 K RAM, novità anteprima inglesi.
Vistoli "Pico Tronic" G. Mario - Via V. Peloni, 26 - 51100 Pistoia - Tel. 35713.

Se possiedi uno ZX Sinclair sei nostro amico! Ci potrai richiedere i migliori programmi ai migliori prezzi, libri inglesi ecc. agli iscritti gli adesivi del club "Gruppo utilizzatori computer Sinclair" c/o.
Chimenti Roberto - Via Luigi Rizzo, 18 - 80124 Napoli - Tel. 081/617368.

Vendo per ZX Spectrum un programma per ridurre sistemi per il totocalcio adattabili al Totip ed Enalotto. Per informazioni rivolgersi ore pasti a:
Rosiello Marcello - Via Cancello Rotto, 10 - 70125 Bari - Tel. 411841.

Cerco computer ZX 81 usato o nuovo 16K + alimentatore 0,7A cavetti di connessione manuale italiano + eventuali listati o programmi cassetta a non più di L. 100.000. Telefonare dopo le 21.00.
Benetti Andrea - Via Cairoli, 4 - 13011 Borgosesia (VC) - Tel. 25062.

Vendo ZX 81 + 16K + alimentatore (il tutto nuovo e garanzia da spedire) + manuali inglese e italiano + cassetta che contiene favolosi programmi (Pacman, Madkong ...) + listati vari, il tutto vendo a L. 345.000. Gli interessati si rivolgano a:
Pintus David - Via Nuoro, 3 - 09042 Mandas (CA) - Tel. 070/984068.

Cerco nella zona di Piacenza possessori di ZX Spectrum per scambio programmi informazioni e consigli.
Affatticati Alessandro - Via S. Franca, 60/C - 29100 Piacenza - Tel. 0523/32953.

Super programmi Spectrum scambiasi!! Rispondo a tutti gli Spectpatiti. Mandatemi la vostra lista ed io invierò la mia per iniziare a scambiarci i programmi intervenite numerosi scrivendo a:
Bottoni Fabio - Via Pr. Uglietti, 16 - 28067 Pernate (NO)

Vendo o cambio software su cassetta per ZX Spectrum.
Bozzoni Giacomo - Via Sardegna, 80 - 20099 Sesto San Giovanni (MI) - Tel. 02/2479771.

Cambio programmi per ZX 81 gestioni, giochi, utilità, inviare lista, risposta assicurata.
Vescovini Giuseppe - Via Flemine, 2 - 41100 Modena - Tel. 352897.

Vendo programmi per ZX Spectrum 48K compilatore Basic L. 30.000 Flight Simulation - Time Gate - Defender ect. L. 12.000 cadauno.
Carletti Mauro - Via dei Cavalieri, 67 - 47037 Rimini (FO) - Tel. 0541/22775.

Cerco possessori ZX Spectrum zona Bari per scambio programmi informazioni e idee annuncio sempre valido anche per utenti di tutta Italia. Inviate vostra lista programmi e io invierò a voi la mia.
Sciancalepore Giuseppe - Via P. Emilio, 50 - 70059 Trani (BA) - Tel. 0883/45682.

Vendo i seguenti manuali in inglese: "ZX Spectrum microdrive and interface 1 manual" a L. 12.000 e "Spectrum microdrive book" Byian Logan, a L. 15.000.
Vialetto Dante - Via Gorizia, 5 - 21053 Castellanza (VA) - Tel. 0331/500713.

Sinclair ZX 81 + 16 K RAM al miglior offerente completo di manuali cavi e alimentatore.
Sartori Carlo - Via Adige, 3 - 31027 Spresiano (TV).

Vendo, causa regalo altro sistema, ZX81 con espansione 32K RAM e con alimentatore potenziato, il tutto in imballi originali con cavi di collegamento. Regalo libro programmi + hardware + manuali + manuale con il sistema + operativo disassemblato e commentato + numerosi programmi su cassetta C90. Il tutto a sole L. 300.000 causa assoluto bisogno di contanti.
Fumagalli Davide - Via Osoppo, 11 - 20148 Milano - Tel. 4033650.

Vendo e scambio oltre 150 programmi per ZX Spectrum a L. 5.000 cad.
Per richiesta listino inviare L. 1.000 in francobolli a:
Pezzali Stefano - Via Loria, 4 - 46100 Mantova - Tel. 0376/362180.

Vendo per ZX Spectrum Routine 2 K per usare 64 colonne sulla stampante e sul video. Vendo anche vari altri programmi soprattutto matematici.
Pepe Emilio - Via R. Montuoro, 5 - 90145 Palermo - Tel. 569141.

Vendo e scambio oltre 150 programmi per ZX Spectrum a L. 5.000 cad. per richiesta listino inviare L. 1.000 in francobolli a:
Pezzali Stefano - Via Loria, 4 - 46100 Mantova - Tel. 0376/362180.

Vendo per ZX Spectrum: compilatore Pascal L. 50.000; Forth L. 20.000; Editor/Assembler L. 15.000 (tutti completi di manuale). Solo zona Roma. Telefonare ore serali.
Frezza Federico - Via Palmi, 1 - 00182 Roma - Tel. 7551195.

Vendo ZX81 16K RAM cavi e alimentatore del 1983 febbraio e cassetta scacchi a L. 200.000.
Ghidoni Elio - Via Venezia, 6 - 25073 Bovezzo (BS) - Tel. 030/2712659.

PICCOLI ANNUNCI

Eccezionale! Vendo cassetta con 2 magnifici giochi per Sinclair ZX Spectrum con logica portante realizzata completamente in linguaggio macchina. Scrivere o telefonare dalle 20 alle 21.
Gatti Alfredo - Via Alzaia, 59 - 27100 Pavia - Tel. 0382/467300.

Vendo/cambio software per Spectrum novità mensili, prezzi eccezionali, richiedetemi gratis l'elenco con più di 120 titoli in continua espansione, vendo inoltre Spectrum 48K a L. 460.000 come nuovo.
Prignano Gianni - Via Portuense, 1450 - 00050 Ponte Galleria, Roma - Tel. 06/6471026.

Eccezionale offerta di programmi per lo Spectrum. Vendita in piccoli e grandi stock di cassette a prezzi convenientissimi. Contattare per ulteriori informazioni.
Vita Luciano - Via Oreste Pennati, 1 - 20052 Monza (MI) - Tel. 039/367029.

Cambio/vendo programmi per ZX Spectrum L. 8.000. Scrivere o telefonare (di sera) per ricevere l'elenco completo.
Calcaterra Stefano - Via Marconi, 34/2 - 40122 Bologna - Tel. 051/521063.

Texas

Vendo stampante PC-100A della Texas, causa passaggio a sistema superiore, in perfette condizioni a L. 350.000 trattabili. Tratto esclusivamente con Genova. Telefonare (circa ore 21) solo se veramente interessati.
Ramasco Luca - Via Miramare, 2 - 16128 Genova - Tel. 587321.

Vendo Texas I 59 stampante PC 100 + schedine magnetiche + modulo di base con relative istruzioni + vasta biblioteca di programmi per ingegneria civile con manuali d'uso L. 380.000 trattabili.
Tavella Giuseppe - P.zza Municipio - 88010 Jonadi (CZ) - Tel. 0963/331124.

Texas TI99/4A cerco programmi semplici adatti a principiante non giochi ma di archivio ecc.
Brunetti Angelo - Via Cherso 3/B - 10136 Torino - Tel. 393695.

Vendo software per TI99 in Ext. Basic: Asteroid, Adventure, Voyage, Rain, Xevios, Abyss e molti altri di ogni genere, tutti originali, a richiesta in Basic. Scrivere per catalogo unendo L. 500 in francobolli.
Prochet Alfredo - Via Dandolo, 19 - 10137 Torino - Tel. 011/304196.

Vendo TI-99/4D + alimentatore + cavi di collegamento registratore e TV + modulo PAL color + modulo SSS minimemori + coppia joystick + manuali d'uso, il tutto in perfette condizioni a L. 500.000.
Biondi Fabio - Via Città Vecchia, 46 - 59049 Vipiteno (BZ).

Vendo 45 programmi per TI 99/4A; dispongo di labirinto 3D, Sci, Tron, Othello, Alien attack, automobilismo 3D, ecc. vendita contrassegno.
Richiedere elenco dettagliato allegando L. 1.500 per fotocopie, prezzi sotto L. 9.000.
Benatti Paolo - Via A. Doria, 25 - 37138 Verona - Tel. 045/560930.

Vendo a prezzi irrisori software per TI99, giochi, matematica, sistemistica, varie, spedire francobollo per risposta immediata e listino prezzi programmi ottimizzati.
Zupi Valerio - Via Traversa Bernardo Quaranta, 9 - 80146 Napoli - Tel. 7521587.

Per TI 99/4A vendo ottimo programma TI-Basic per grafici singolo Pixel (128x96) o (160x96) esecuzione <3 min. contatterei altri utenti per computer club, in zona.
Rolandi Franco - C. Tassoni, 47 - 10143 Torino - Tel. 011/766724.

Vendo TI 99/4A - CPU 16 bit, 16 K RAM, 26 K ROM, grafica 192x256 punti 16 colori, 3 generatori sonori indipendenti + cavo registratore + joystick + trasformatore, tutto in garanzia a un ottimo prezzo.
Galgani Danilo - P.zza Mattirolo, 8 - 10149 Torino - Tel. 298053.

Per i possessori del TI-99/4A vendo 30 programmi (giochi, matematica, musica, totocalcio) di cui molti originali Texas a L. 50.000 per dettagliate informazioni o lista software contattare:
Santangelo Enzo c/o Casa Universitaria Bertoni - Via Carnia, 14 - 20132 Milano - Tel. 02/2856924.

Varie

Vendo computer N.E. 32K RAM, completo di tastiere, interfacce video e cassette, alimentatore, ventola, il tutto in un rack. Regalo il mobile e i bollettini del club prezzo trattabilissimo.
Bianchi Marco - Via Montello, 23 - 40131 Bologna - Tel. 051/557265.

Sono interessato a tutto il software disponibile per "Oric" chi ne fosse in possesso prego scrivere o telefonare a.
Buratti Giacomo - Via Metastasio, 4 - 20098 S. Giuliano M.se (MI) - Tel. 9844350.

Vendo videogiochi Philips con 6 cassette + musicale a L. 350.000 spedizione compresa.
Vidili Vincenzo - C.so Regio Parco, 122 - 10154 Torino - Tel. 011/267249.

Vendo software per Olivetti M20, con manuali richiedete catalogo gratis. Prezzi molto convenienti.
Occhetta Elsa - Via Beldì, 19 - 28068 Romentino (NO).

Scambio programmi Apple II. Compero inoltre dischetto system master purchè vera occasione. Vendo inoltre il manuale Sinclair "Microdrive and interface 1 manual" a L. 12.000.
Vialetto Dante - Via Gorizia, 5 - 21053 Castellanza (VA) - Tel. 0331/500713.

Sono in contatto con alcuni TRS-80ntisti. Allarghiamo il giro di amici o scambio notizie e programmi. Scrivete risponderò a tutti.
Politi Massimo - Via Gabriele D'annunzio, 31 - 65100 Pescara - Tel. 085/690786.

Vendo HP41C + lettore di schede + modulo memoria quadruplo + modulo matematica + batterie ricaricabili + carica batterie + libri, schede e programmi vari, tutto come nuovo a L. 700.000 trattabili.
Fasano Amedeo - Tel. 02/2130331.

Vendo programmi di ingegneria civile, telai, pendii, plinti, travi, pilastri, reti idriche, impianti solari ecc. ecc. a prezzi molto interessanti - parecchi manuali - richiedere elenco.
De Gioanni Maurizio - C.so S. Santanosa, 67 - 12100 Cuneo - Tel. 017/61839.

Vendo computer NEZ80 espanso 60k con scheda videografica, monitor verde 12", un drive tandom. Il tutto racchiuso in due mobili rack. Dispongo delle schede LX383, LX384, LX385, LX386, LX388, LX389.
Feroldi Maurizio - Via Serio, 30 - 26100 Cremona - Tel. 0372/411672.

Vendo o cambio micro Z80 NE con le seguenti schede complete LX382, 383, 384, 385, 388, 389, 390, 392 monitor FV12 mobile L. 600.000 opp. cambio con ZX81 64K e digital Ker o con stampante 80 coc centronics.
Vonato Massimo - Via XX Settembre, 32 - 28010 Gargallo (NO) - Tel. 0322/355042.

Per Sharp MZ-80K vendo bellissimi giochi di animazione, lista di indirizzi di utilissimi poke, implementazioni del Basic, Pascal, System programs, word processing, data base e molto altro.
Giovanelli Claudio - Via Ripamonti, 194 - 20141 Milano - Tel. 02/536926.

Vendesi Sharp 1500 + stampante 4 colori CE150 + espansione 5K L. 750.000 completo manuali. Registratore digitale Philips D6600 L. 80.000. Telefonare ore 20.
Del Vecchio Francesco - Via Amoruso, 34 - 70124 Bari - Tel. 510322.

Software per Apple II - Visicalc L. 150.000; Visiterm L. 120.000; Visiplot L. 100.000; Visidex L. 100.000; Desktop-plan II L. 100.000; Visidrent + visiplot L. 120.000.
Lenzi Paolo - Via Desenzano, 20146 Milano - Tel. 4040574.

Vendo interfaccia Apple IEEE 488 a L. 500.000. Vendo interfaccia parallela centronics a L. 80.000. Vendo interfaccia seriale RS 232 C a L. 140.000.
Lenzi Paolo - Via Desenzano - 20146 Milano - Tel. 4040574.

Vendo interfaccia doppio floppy disk per Apple II o Lemon a L. 100.000. Vendo scheda orologio/calendario al Quarzo "Mountain hardware" per Apple II a L. 500.000.
Lenzi Paolo - Via Desenzano - 20146 Milano - Tel. 4040579.

Acquisto libri e riviste di carattere generale e/o dedicate al TRS-80. Cerco numeri anche sciolti di "80 microcomputing" e "Creative computing".
Politi Massimo - Via Gabriele D'Annunzio, 31 - 65100 Pescara (PE) - Tel. 085/690786.

CEDOLA DI COMMISSIONE LIBRARIA PERSONAL SOFTWARE

Da inviare a Gruppo Editoriale Jackson - Via Rosellini, 12 - 20124 Milano

Nome Cognome

Indirizzo

Cap. Città Provincia

Partita I.V.A. (indispensabile per le aziende)

Si richiede l'emissione della fattura

Inviatemi i seguenti libri:

Codice Libro	Quantità	Codice Libro	Quantità	Codice Libro	Quantità	Codice Libro	Quantità	Codice Libro	Quantità

□ Pagherò al postino il prezzo indicato + L. 2.000 per contributo fisso spese di spedizione.

□ Allego assegno n° .... di L. ....

□ Non abbonato □ Abbonato sconto 20% □ l'Elettronica □ Elettronica Oggi □ Automazione Oggi □ Elektor □ Informatica Oggi □ Computerworld □ Bit □ Personal Software □ Strumenti Musicali □ Videogiochi

Data .... Firma ....

PICCOLI ANNUNCI PERSONAL SOFTWARE

Sei un lettore di PERSONAL-SOFTWARE e vuoi entrare in contatto con tutti gli altri lettori per comperare, cambiare o vendere software? Spedisci questo tagliando a Gruppo Editoriale Jackson - Via Rosellini, 12 - 20124 Milano

Nome .... Cognome ....

Via .... C.A.P. ....

Città .... Tel. ....

UNA PUBBLICAZIONE
DEL GRUPPO EDITORIALE JACKSON

PERSONAL SOFTWARE

ANNO 3 N. 14 GENNAIO 1984

DIRETTORE RESPONSABILE: Giampietro Zanga

DIRETTORE: Pietro Dell'Orco

REDAZIONE: Lorenzo Barrile

COORDINAMENTO: Riccardo Paolillo

CONSULENZA: Giacomo Bortone

HANNO COLLABORATO A QUESTO NUMERO:
F. Stella, R. Mazzurco, M. Flora, A. Guida, S. Orlando, F. Carbone, M. Coccorese, S. Borsani, S. Cerutti, C. Driussi, A. Motta, F. Sardo, T. Policastro, L. Iacono, M. Spero, M. Lenzi

GRAFICA E IMPAGINAZIONE: Luigi Chiesa

PUBBLICITA': Concessionario per l'Italia e l'Estero
Reina S.r.l. Via Washington, 50 - 20146 Milano
Tel. (02) 4988066/7/8/9/060 (5 linee r.a.)
Telex 316213 REINA I

FOTOCOMPOSIZIONE: LINEACOMP S.r.l.
Via Rosellini, 12 - 20124 Milano

STAMPA: REWEBA - Brescia

Concessionario esclusivo per la diffusione in Italia e all'Estero:
SODIP - Via Zuretti, 25 - 20125 Milano

AUTORIZZAZIONE ALLA PUBBLICAZIONE:
Tribunale di Milano n. 69 del 20/2/1982

Spedizione in abbonamento postale Gruppo III/70
Prezzo della rivista L. 3.500. Numero arretrato L. 7.000.
Abbonamento annuo (10 numeri) L. 28.000; per l'Estero L. 44.800

I versamenti vanno indirizzati a: Gruppo Editoriale Jackson Via Rosellini, 12 - 20124 Milano - mediante emissione di assegno bancario, cartolina vaglia o utilizzando il C/C postale numero 11666203.
Per i cambi di indirizzo, indicare, oltre naturalmente al nuovo, anche l'indirizzo precedente, ed allegare alla comunicazione l'importo di L. 500, anche in francobolli.

Il Gruppo Editoriale Jackson è iscritto nel registro Nazionale della stampa al n. 117 - Vol. 2 - Foglio 129 in data 17-8-1982

GRUPPO EDITORIALE JACKSON S.r.l.

DIREZIONE, REDAZIONE, AMMINISTRAZIONE:
Via Rosellini, 12 - 20124 Milano - Telefoni: 68.80.951/2/3/4/5

SEDE LEGALE: Via Vincenzo Monti, 15 - 20123 Milano

DIREZIONE EDITORIALE: Giampietro Zanga e Paolo Reina

COORDINAMENTO EDITORIALE: Daniele Comboni

DIREZIONE AMMINISTRATIVA: Giuliano Di Chiano

Esposizioni Internazionali dell'Automazione

...1982 Parigi "MESUCORA"... 1983 Düsseldorf "INTERKAMA"

1984 MILANO - B.I.A.S.

Solo il BIAS nel 1984 in Europa presenta l'Automazione e la Microelettronica



studio martinetti

19° Convegno Mostra Internazionale dell'Automazione Strumentazione e Microelettronica

Fiera di Milano
29 novembre - 4 dicembre 1984

- Sistemi e Strumentazione per l'Automazione la regolazione ed il controllo dei processi Robotica, sensori e rilevatori
- Apparecchiature e Strumentazione per laboratorio, collaudo e produzione
- Componentistica, sottoassiemi periferiche ed unità di elaborazione
- Micro, Personal Computer, Software e accessori

E.I.O.M. Ente Italiano Organizzazione Mostre
Segreteria della Mostra
Viale Premuda 2
20129 Milano
tel. (02) 796096/421/635 - telex 334022 CONSEL

in concomitanza con la 8ª RICH e MAC '84

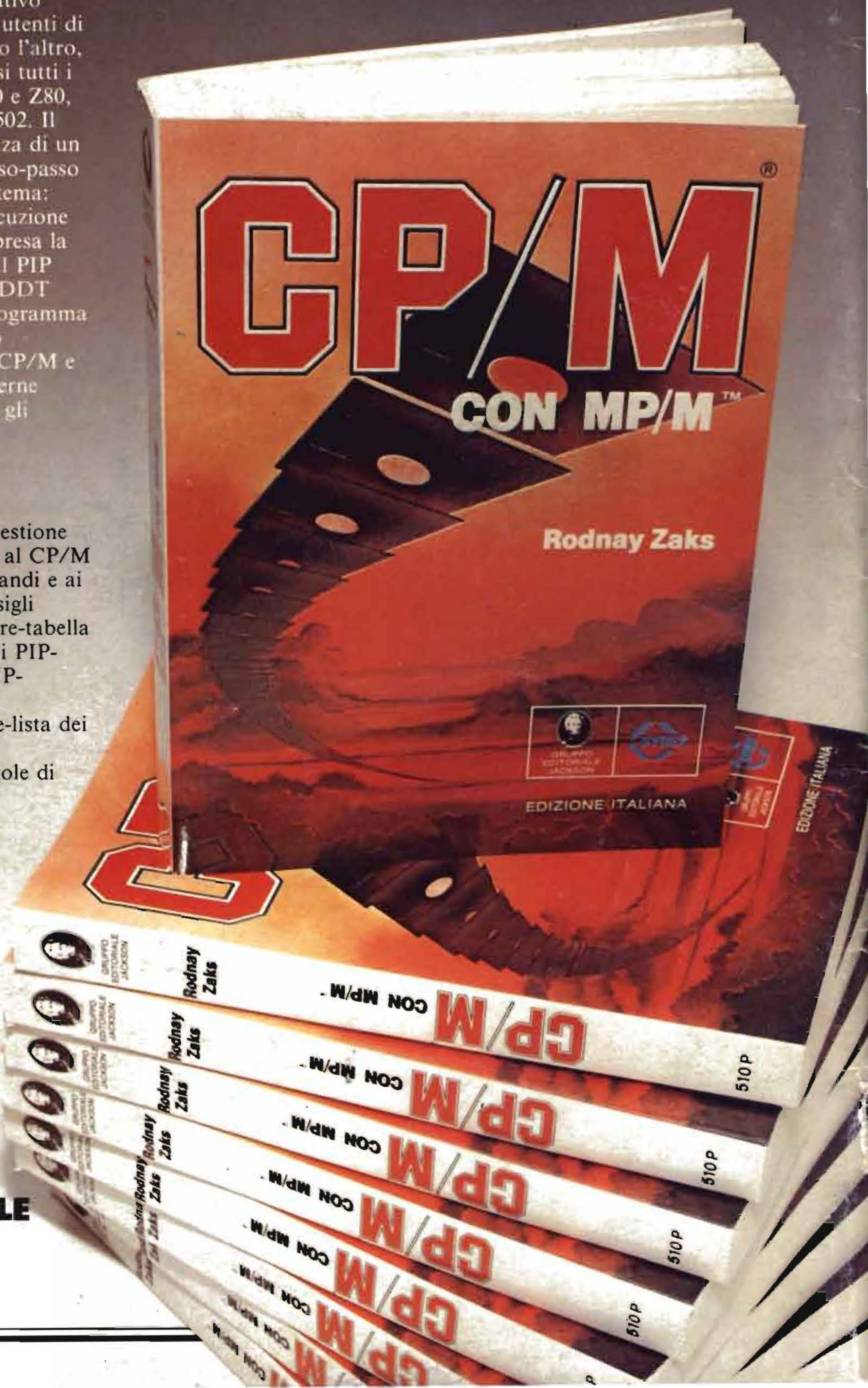
Usare il sistema operativo CP/M

IL LIBRO

Il sistema operativo CP/M è stato progettato per rendere semplice l'uso di un microcomputer. Questo libro vi renderà semplice l'uso del CP/M. (Le versioni esaminate del CP/M sono il CP/M 1.4-il CP/M 2.2. e il nuovo sistema operativo multiutente MP/M) La maggior parte di utenti di microcomputer dovrà, infatti, un giorno o l'altro, fare ricorso al CP/M, disponibile su quasi tutti i computer basati sui microprocessori 8080 e Z80, come pure su certi sistemi utilizzanti il 6502. Il libro, senza presupporre alcuna conoscenza di un calcolatore, inizia con la descrizione, passo-passo delle procedure di inizializzazione del sistema: accensione, inserimento dei dischetti, esecuzione delle più comuni operazioni su file, compresa la duplicazione dei dischetti. Prosegue con il PIP (programma di trasferimento dei file), il DDT (programma di messa a punto) e ED (programma editor). Per entrare sempre più, fornendo numerosi consigli pratici, all'interno del CP/M e delle sue operazioni, al fine di comprenderne appieno le risorse ed eventualmente dare gli strumenti per successive modifiche.

SOMMARIO

Introduzione al CP/M e all'MP/M-Le caratteristiche del CP/M e dell'MP/M-Gestione dei file con PIP-L'uso dell'editor-Dentro al CP/M e all'MP/M-Guida di riferimento ai comandi e ai programmi del CP/M e dell'MP/M-Consigli pratici-Il futuro-messaggi comuni di errore-tabella di controllo di ED-nomi dei dispositivi di PIP-riassunti dei comandi-parole chiave di PIP-parametri di PIP-tasti di controllo per la digitazione dei comandi-tipi di estensione-lista dei materiali-organizzazione della stanza del calcolatore-verifiche in caso di errore-regole di base per la localizzazione dei guasti.



Pagg. 320 Cod. 510P
L.22.000 (Abb. L.19.800)

SCONTO 20% AGLI ABBONATI
FINO AL 28-2-'84

Per ordinare il volume utilizzare l'apposito tagliando inserito in fondo alla rivista.

GRUPPO EDITORIALE JACKSON
Divisione Libri